

**TRANSPORTATION ANALYSIS SIMULATION SYSTEM
(TRANSIMS)**

Version: TRANSIMS-LANL-1.0

**VOLUME 2 – SOFTWARE
PART 1 – MODULES**

28 May 1999

LA-UR 99-2574

COPYRIGHT, 1999, THE REGENTS OF THE UNIVERSITY OF CALIFORNIA. THIS SOFTWARE WAS PRODUCED UNDER A U.S. GOVERNMENT CONTRACT (W-7405-ENG-36) BY LOS ALAMOS NATIONAL LABORATORY, WHICH IS OPERATED BY THE UNIVERSITY OF CALIFORNIA FOR THE U.S. DEPARTMENT OF ENERGY. THE U.S. GOVERNMENT IS LICENSED TO USE, REPRODUCE, AND DISTRIBUTE THIS SOFTWARE. NEITHER THE GOVERNMENT NOR THE UNIVERSITY MAKES ANY WARRANTY, EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY OR RESPONSIBILITY FOR THE USE OF THIS SOFTWARE.

TRANSIMS

Version: TRANSIMS-LANL-1.0

VOLUME 2 – SOFTWARE PART 1 – MODULES

28 May 1999

LA-UR 99-2574

The following persons contributed to this document:

C. L. Barrett*
R. J. Beckman*
K. P. Berkbigger*
K. R. Bisset*
B. W. Bush*
S. Eubank*
J. M. Hurford*
G. Konjevod*
D. A. Kubicek*
M. V. Marathe*
J. D. Morgeson*
M. Rickert*
P. R. Romero*
L. L. Smith*
M. P. Speckman**
P. L. Speckman**
P. E. Stretz*
G. L. Thayer*
M. D. Williams*

* Los Alamos National Laboratory, Los Alamos, NM 87545

** National Institute of Statistical Sciences, Research Triangle Park, NC

Acknowledgments

This work was supported by the U. S. Department of Transportation (Assistant Secretary for Transportation Policy, Federal Highway Administration, Federal Transit Administration), the U. S. Environmental Protection Agency, and the U. S. Department of Energy as part of the Travel Model Improvement Program.

CONTENTS

1. POPULATION SYNTHESIZER MODULE.....	7
1.1 OVERVIEW	7
1.2 ALGORITHM.....	8
1.3 USAGE	16
1.4 TUTORIAL	21
1.5 TROUBLESHOOTING	53
1.6 SIMPLIFIED POPULATION GENERATOR.....	53
2. ACTIVITY GENERATOR MODULE	56
2.1 OVERVIEW	56
2.2 ALGORITHM.....	56
2.3 USAGE	67
2.4 TUTORIAL	68
2.5 TROUBLESHOOTING	68
2.6 FILES	68
2.7 SIMPLIFIED ACTIVITY GENERATOR.....	75
3. ROUTE PLANNER MODULE	80
3.1 OVERVIEW	80
3.2 ALGORITHM.....	81
3.3 USAGE	89
3.4 TROUBLESHOOTING	92
4. TRAFFIC MICROSIMULATOR MODULE.....	93
4.1 OVERVIEW	93
4.2 ALGORITHM.....	94
4.3 USAGE	115
4.4 TUTORIAL	119
4.5 TROUBLESHOOTING	120
5. EMISSIONS ESTIMATOR MODULE	122
5.1 OVERVIEW	122
5.2 LIGHT-DUTY VEHICLE TAILPIPE SUBMODULE	122
5.3 ALGORITHM.....	131
5.4 USAGE	132
5.5 TUTORIAL	134
5.6 TROUBLESHOOTING	135
5.7 REFERENCES	139
6. OUTPUT VISUALIZER MODULE.....	140
6.1 OVERVIEW	140
6.2 REQUIREMENTS	140
6.3 TUTORIAL	141
6.4 CURRENT COLORMAP SETTINGS	160
6.5 UTILITY PROGRAMS	162

6.6 TROUBLESHOOTING	164
---------------------------	-----

1. POPULATION SYNTHESIZER MODULE

1.1 Overview

TRANSIMS is based on the movements of individual travelers between activities at different locations. This makes necessary the creation of a TRANSIMS synthetic population that is a representation of every household and person in a metropolitan region being studied. This representation of households and persons includes their demographics and where they live. In TRANSIMS, travel is induced when a person engages in an activity at one location and time and must partake in another activity at a different location later in the day. The activities one engages in and their locations are often driven by household and personal demographics. For example, demographics such as the age of the individual, his/her income, gender, and whether he/she is employed greatly influence the person's daily itinerary. Therefore, in the methodology outlined here, not only are the individual households generated, but the household and individual demographics are also estimated.

A baseline synthetic population is generated using three products from the Census Bureau. The first is Standard Tape File-3A, STF-3A (Census, 1992a). This file contains demographic summary tables from the 1990 Census for the small geographic areas, census tracts, or census block groups. The summary tables, usually one-dimensional, contain such things as the distribution of the age of the householder or the number of workers in the family. For example the STF-3A tables summarizing the number of workers in families and the age distribution of the householder in family households in census tract 1, block group 2 of Los Alamos County, New Mexico are given in Table 1 and Table 2:

Table 1: The number of workers in family households for census tract 1, block group 2 of Los Alamos County, NM.

Number of Family Households, n , with Number of Workers in the Household				
Workers	0	1	2	>2
n	0	121	214	25

Table 2: The age distribution of householders for census tract 1, block group 2 of Los Alamos County, NM.

Number of Family Households, n , with Householder Age in the Given Ranges							
Age	15-24	25-34	35-44	45-54	55-64	65-74	>74
n	4	134	94	46	46	36	0

Census tracts and block groups are combined into much larger geographic areas called Public Use Microdata Areas, PUMA. For each PUMA the Census Bureau has available a 5% sample of the complete census records for households in the area, sans any identification such as the name, address, block group, or census tract of the household. This set of data is called the Public Use Microdata Sample, PUMS (Census, 1992b). The sample records of the PUMS contain the complete structure of the households. That is, all of the demographics collected by the Census

Bureau for a household are given. This includes the number of people in the household, the household income, the number of workers in the household, and the number of vehicles owned by the household members. Demographics for each household member are also given. Some of these are the age of the person, the gender, whether the person is employed or in school, and the income.

The last set of data used in the creation of the baseline population is the MABLE'98/Geocorr v3.0 search engine and data that is maintained by the center for International Earth Science Information Network of Columbia University, CIESIN, a non-profit, non-government organization. It is available at the web site, <http://plue.sedac.ciesin.org/plue/geocorr>. This search engine and combined data set produce a correspondence between PUMAs and the census block groups.

Using these three data sets, a synthetic population is generated according to the algorithm developed by Beckman, Baggerly, and McKay [1996]. Individual households are placed at activity locations on the transportation network using land-use data. The complete algorithm is described in the following section.

1.2 Algorithm

The data flow for creating and locating the synthetic population is given in Figure 1.

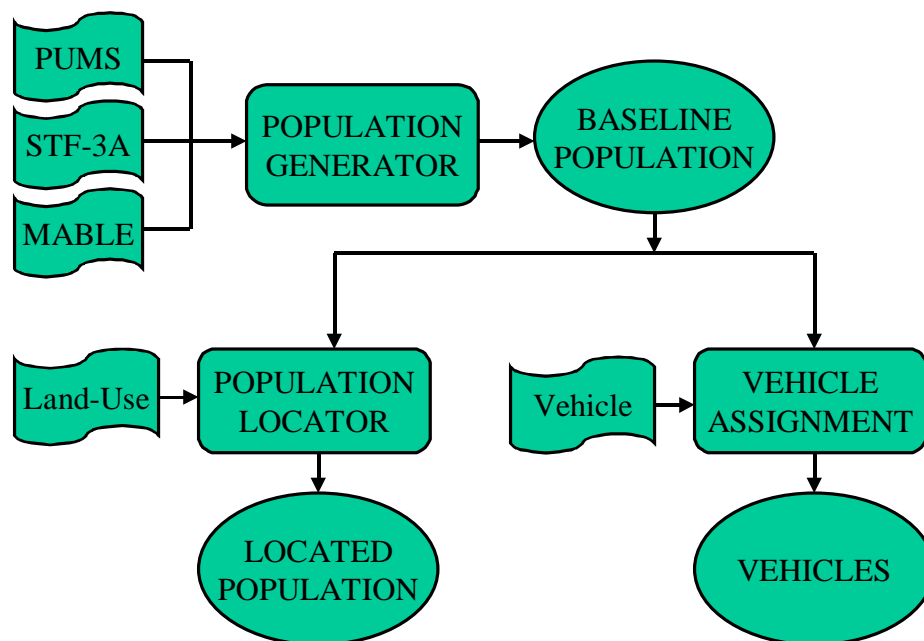


Figure 1: A flowchart of the process to create, locate, and assign vehicles to a complete synthetic population of households and individuals. PUMS, STF-3A, MABLE/Geocorr, Land-Use and Vehicle data are external data sets used in this process.

The three files—PUMS, STF-3A, and the MABLE-Geocorr—are inputs to the Population Synthesizer that produces a baseline synthetic population. At this stage, the households are known to reside in a block group, but their exact location within the block group is not known. Land-use

data is used to refine these locations resulting in each household being located at one activity location on a network link. Each household and individual in the population is given a unique identification number in this process.

The number of vehicles each household possesses is generated by the Population Synthesizer. The vehicles are assigned a type using external data such as the national distribution of vehicle types. The output from this process is the vehicle file that gives the ownership of the vehicles in addition to their type.

1.2.1 Introductory Techniques

Of the processes shown in Figure 1, the baseline Population Synthesizer is by far the most complex and is considered first. The objective is to create a population over the small geographic areas of census block groups that maintains the statistical characteristics of the Census. However, as is shown in Table 3, the summary data for block groups from STF-3A does not give the entries for any of the cross-classified demographics.

Table 3: The cross-classification of the number of workers and the age of the householder for census tract 1, block group 2 of Los Alamos County, NM is unknown.

Workers	Householder Age							Total
	15_24	25-34	35-44	45-54	55-64	65-74	>74	
0	?	?	?	?	?	?	?	0
1	?	?	?	?	?	?	?	121
2	?	?	?	?	?	?	?	214
>2	?	?	?	?	?	?	?	25
Total	r	134	94	46	46	36	0	

If cross-classified tables existed for small areas such as block groups, a synthetic population could be easily generated. Since the PUMS contains complete household records, these could be drawn at random to satisfy the cross-classified table for the block group. This is the general scheme for the algorithm given here, except that the cross-classified table for the block groups is estimated. This estimation process satisfies the totals as given by STF-3A. The general methodology is:

- 1) Select a reasonable set of demographics from STF-3A that characterize the population.
- 2) For each block group, estimate the proportions in the cross-classified table made up of the demographics selected in step 1.
- 3) Draw households at random from the PUMS corresponding to the block group so that the estimated proportions in the cross-classified table are satisfied.

While cross-classified tables cannot be derived from STF-3A for small areas like block groups, multiway summary tables for the entire PUMA area can be created. For example, block group 2 of census tract 1 for Los Alamos County, NM is contained in PUMA 00400. The multiway table for this PUMA showing the number of workers in a family and the age of the householder is given in Table 4.

Table 4: The cross-classification of the number of workers and the age of the householder for PUMA 00400, which contains census tract 1, block group 2 of Los Alamos County, NM.

Householder Age								
Workers	15-24	25-34	35-44	45-54	55-64	65-74	>74	Total
0	2	11	9	3	26	64	42	157
1	11	108	122	48	80	61	18	448
2	28	135	274	156	85	22	6	706
>2	0	3	65	76	40	10	3	197
Total	41	257	470	283	231	157	69	

The proportions in the cells of the multiway block group tables are estimated using iterative proportional fitting [Deming and Stephan, 1940], IPF, of the block group summaries to the cross-classified values in the PUMS. The assumption throughout this paper is that the correlation structure of the demographics for every entity that contributes to the PUMA (e.g., block groups) is same as the correlation structure in the multiway tables constructed from the PUMS. IPF assures that this is true.

IPF assumes that we have a sample from a multiway classification of characteristics and the exact totals for the margins of the multiway table. In this case, we could assume that the PUMS represents the sample and the STF-3A data gives the marginal totals. We show later that this is an over-simplified view of these data, but continue with this to better explain IPF.

IPF estimates (refines) the entries in the sample multiway table (here the PUMS) to make them exactly match the known marginals (here the STF-3A summary data) while maintaining the correlation structure of the sample table. The algorithm is exceedingly simple. First, all summaries and tables are converted to proportions of the total. For example Table 1 and Table 2 above become Table 5 and Table 6.

Table 5: The proportion of workers in family households for census tract 1, block 2 of Los Alamos County, NM

Proportion of Family Households, n, with Number of Workers in the Household				
Workers	0	1	2	>2
Prop.	0.000	0.336	0.594	0.069

Table 6: The proportion of ages of householders for census tract 1, block group 2 of Los Alamos County, NM.

Proportion of Family Households, n, with Householder Age in the Given Ranges							
Age	15-24	25-34	35-44	45-54	55-64	65-74	>74
Prop.	0.011	0.372	0.261	0.128	0.128	0.100	0.000

In terms of proportions, the PUMS sample for these two demographics is shown in Table 7.

Table 7: The cross-classification of the proportion of workers and the age of the householder for PUMA 00400, which contains census tract 1, block group 2 of Los Alamos County, NM.

Householder Age								
-----------------	--	--	--	--	--	--	--	--

Workers	15-24	25-34	35-44	45-54	55-64	65-74	>74	Total
0	0.001	0.007	0.006	0.002	0.017	0.042	0.028	0.104
1	0.077	0.072	0.081	0.032	0.053	0.040	0.012	0.297
2	0.019	0.090	0.182	0.103	0.056	0.015	0.004	0.468
>2	0.000	0.002	0.043	0.050	0.027	0.007	0.002	0.131
Total	0.027	0.170	0.312	0.188	0.153	0.104	0.046	

The PUMS proportions as shown in Table 7 are converted by IPF to have the same row and column proportions as the STF-3A data given in Table 5 and Table 6 by first changing the rows of the table and then changing the columns according to the following rules: Update the first row of Table 7 by multiplying each entry by the first marginal proportion for that row given in Table 5 and dividing by the total for that row on the last iteration. In this case, the first element of the first row of Table 7 becomes $0.001 * 0.000 / 0.104 = 0.000$. This process continues with the remainder of the rows of Table 7, where, for example, the third entry of the second row becomes $0.081 * 0.336 / 0.297 = 0.092$. After all the rows of Table 7 are updated, the exact same procedure is applied to each of the columns of the table. The procedure continues by alternating between the rows and columns of the table until the table entries are no longer changing. For tables with more than two dimensions, the same procedure is followed—updating one dimension at a time. The final result of IPF to Table 7 is given in Table 8.

Table 8: The estimated cross-classification of the proportion of workers and the age of the householder for families in census tract 1, block group 2 of Los Alamos County, NM.

Householder Age								
Workers	15-24	25-34	35-44	45-54	55-64	65-74	>74	Total
0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1	0.003	0.141	0.061	0.020	0.047	0.063	0.000	0.336
2	0.009	0.228	0.178	0.086	0.065	0.030	0.000	0.594
>2	0.000	0.003	0.022	0.022	0.016	0.007	0.000	0.069
Total	0.011	0.372	0.261	0.128	0.128	0.100	0.000	

The last step in household generation is to draw samples from the PUMS. There are 360 family households in the block group given above. For this block group, 360 households are generated—one at a time—following this procedure: First, a category of age and the number of workers is chosen at random according to the probabilities in Table 8. Then, given the category, say a householder between 45 and 54 years of age in a household with two workers, one of the households in the PUMS matching these demographics is drawn at random. In this case, one household would be drawn from the 156 households possible as shown in Table 4. This process is repeated 360 times to form a population that matches the Census. It should be noted that the same household from the PUMS may be chosen multiple times by this procedure.

It is shown in the paper by Beckman, Baggerly and McKay [1996], and it is obvious from Tables 1, 2 and 4 that the procedure of fitting only one block group at a time using IPF is not entirely correct. IPF is based on the assumption that the *seed* proportions as given by the PUMS, Table 7, are a sample of the population that produces the exact marginal totals give by STF-3A and shown here in Table 1 and Table 2. The PUMS is a sample of households that contain all or parts of multiple block groups. In the case shown here, block group 2 of census tract 1 from Los Alamos

County, NM, is just one of the many block groups in PUMA 00400. In actual fact, PUMA 00400 contains all of the block groups in Los Alamos and Santa Fe Counties of New Mexico. That PUMA 00400 is a sample of multiple block groups is evident from Table 1, which states that there are no households with zero workers, while the sample given by the PUMS shows such households. Therefore, the correct procedure is to consider all of the block groups in the PUMA simultaneously.

1.2.2 Creating the Baseline Population

The true synthetic population is constructed as follows: Each block group in a PUMA is considered. These are assembled from the MABLE/Geocorr database. In some small percentage of the cases, a block group is split between multiple PUMAs. Summary tables for a selected set of demographics from STF-3A are assembled for each of the block groups. In the case of a split block group, the summary totals are reduced by the proportion of the block group's households in the PUMA. This information is also available from the MABLE/Geocorr database. The multiway demographic table, which matches the demographics from the STF-3A tables for the corresponding PUMA is constructed from the PUMS. Then, a multiway table is estimated for each census tract where the marginal totals match those given by STF-3A. The correlation structure in the estimated table is the same as the correlation structure in the multiway table constructed from the PUMS. The last step in the process is the same as before where random households are drawn from the PUMS which *match* the demographics of the each of the cells of the estimated multiway table for each block group.

The simultaneous consideration of all block groups that comprise a PUMA requires that the IPF routine be modified to the following two step procedure: Here the m demographic variables from STF-3A will make up the marginal values of the multiway table. First, the marginal tables for all of the block groups in the PUMA are added. A multiway table for the entire PUMA is estimated by an IPF fit of this summed table to the PUMS. In the second step, this estimated table is used as an additional marginal table. Also, an $(m+1)$ -dimensional table is created. The first m dimensions are the m marginals from STF-3A, while the $m+1^{st}$ marginal is created by *stacking* all of the marginal tables. This $(m+1)$ -dimensional stacked table along with the table estimated from the sums are the marginal tables used in an IPF procedure to an $(m+1)$ -dimensional table consisting entirely of ones. More information on this process can be found in the paper by Beckman, Baggerly and McKay [1996].

Multiple demographics from STF-3A are used to create the synthetic population. Synthetic households are divided into three categories:

- 1) Family households—those households with two or more related persons
- 2) Nonfamily households—persons living alone or unrelated persons living together
- 3) Group quarters—dwellings such as prisons or college dormitories

Since travel activity can depend on the type of household, a synthetic population of households and group quarters is generated for each of the three types. Family households are considered first. The summary tables in STF-3A that concern family households are:

- 1) P24: Age of the Householder
- 2) P107: Family Income
- 3) P112: Number of Workers in the Family

- 4) P124A&B: Poverty Status (which is not used here) * Race * Family Type * Presence and Age of Children

Not all categories that are given for the above STF-3A tables are used in the procedure. For example, there are 25 categories of income in table P107. These are collapsed to only seven categories. The Census STF-3A table P124 (A & B) is used to create a race by family class summary table. Data in the categories of below the poverty level and above the poverty level was added to give the resulting 12 family types. These are:

- 1) Married Couple: Children under age 5 only
- 2) Married Couple: All children between 5 and 17
- 3) Married Couple: Children under 5 and 5 to 17
- 4) Married Couple : No children under 18
- 5) Male Householder-No Wife Present: Children under age 5 only
- 6) Male Householder-No Wife Present: All children between 5 and 17
- 7) Male Householder-No Wife Present: Children under 5 and 5 to 17
- 8) Male Householder-No Wife Present: No children under 18
- 9) Female Householder-No Husband Present: Children under age 5 only
- 10) Female Householder-No Husband Present: All children between 5 and 17
- 11) Female Householder-No Husband Present: Children under 5 and 5 to 17
- 12) Female Householder-No Husband Present: No children under 18

Summary tables in STF-3A for nonfamily households are:

- 1) P17: Household Type and Relationship
- 2) P20: Race * Household Type * Presence and Age of Children. (The race of nonfamily householders can be derived from this table.)
- 3) P24: Age of Nonfamily Householder
- 4) P110: Nonfamily Household Income
- 5) P127: Poverty Status (not used here) * Age of Householder * Household Type

There are only two summary tables for group quarters in STF-3A. These are:

- 1) P40: Group Quarters,
- 2) P41: Group Quarters * Age

Minor adjustments must be made to the IPF routine to handle marginal summaries that are tables. For example, the two-way marginals such as the race * family class table given above are considered to be one marginal by the IPF routine. Such marginal tables are converted to a single demographic whose categories are all the combinations of the two demographics involved. If two marginal tables contain a common demographic variable (e.g., the alone/not alone demographic in tables P17 and P127 in Table 2), the procedure is not altered and the fitting proceeds as above treating each marginal separately. For the case where one demographic variable is in two summary tables (a one-dimensional table and a two-dimensional table) and has fewer categories in the two-dimensional table (e.g., STF-3A summary tables P24 and P127), an additional step is required. The procedure uses only one marginal table at a time. When the table with the *collapsed* marginal is considered, the procedure updates the cells as usual where all of the cells that contribute to the individual *collapsed* categories are updated by the same proportion.

1.2.3 Locating, Numbering the Population and Assigning Vehicle Emissions Type

Three steps remain for creation of the synthetic population. The baseline synthetic population is produced on a block-group basis. No other information about the location of individual households is known. Procedures are developed using land-use data to place each household on the transportation network. The number of vehicles owned by each household is given in the PUMS and is therefore in the synthetic population. The Emissions Estimator module described in Section 5 requires that the vehicles be identified by emissions type. However, the emissions type of the vehicles in the household is unknown. The Population Synthesizer contains a model to assign vehicle types to each vehicle in the population. The last step in the creation of the synthetic population is to assign a unique number to each household and each person in the population.

Each household in the population is located at an activity location. These are usually on the *walk* portion of the network, which is described in Section 7 of Volume 3—*Files*. Each network is required to have an *Activity Location* file. This file contains the locations of those places on the network where activities may take place. Associated with these locations is a set of land-use characteristics that indicate the type of activities that may take place at that location. Each network has a unique set of land-use characteristics associated with its activity locations. The land-use is used to form a weighting factor for each activity location that represents the relative likelihood of a housing unit being placed there. The exact formulation of these weights depends on the network under investigation and the availability of land-use information. For a network representing a real metropolitan area, the land-use could, for example, contain the square footage of single family residential housing along with the square footage of multifamily housing that surrounds the activity location. The weights for the activity locations in this case may be formed by adding the square footage of single family residential housing to a multiple, say 10, of the square footage of multifamily housing. Calibration networks have land use associated with activity locations that is less realistic than housing square footage. The land-use in these networks is used parametrically to study the characteristics of travel on that network. Descriptions of the formation of the housing weights for each calibration network are given in Volume 2—*Files*, Part 3—*Test Networks*.

Given the housing weights associated with each activity location on the network, households are placed on these locations according to the following scheme. First, all activity locations within a block group are identified. Assume that there are n of them. Let the associated household weights for these activity locations be denoted by w_i . The following probabilities are computed from these:

$$p_i = w_i / \sum w_i$$

Now, each individual household in the block group is assigned to one of the n activity locations according to the probabilities, p_i . The location of the households is one of the required demographics for each of the synthetic households. The complete list of required characteristics of the synthetic population and their format may be found in Section 2 of Volume 3—*Files*.

Each synthetic household is created with a number of vehicles assigned to it. These vehicles are given a unique number and identified as belonging to the household. One of the vehicle emission types shown in the Emissions Estimator described in Section 5 is assigned to each vehicle in the population. This assignment is done at random according to a national distribution of vehicle emission types. Additionally, a starting location, one of the parking locations on the driving network, is assigned to each vehicle. Traditionally, this location has been the parking location that

is closest to the household location. This information is written to the vehicle file. Its format may be found in Volume 3—*Files*.

The last step to generate a synthetic population is to assign a unique number to each household and person in the population. The person number is the unique identifier that is carried through the Route Planner to the Traffic Microsimulator. All of the output that is person oriented references these numbers.

Each vehicle driver in TRANSIMS is required to have an entry in the synthetic population. Therefore, fictitious persons are added to the population to represent those who travel on the network but do not live in the area being studied. We call them “itinerant travelers.” Itinerant travelers are added to the synthetic population as single person households each with one vehicle. These households, along with the person, are given their own unique household and person number. If demographics are added to the actual synthetic persons or households, then the same demographics must added to the itinerant traveler population. These may be meaningless numbers, however, as the activity list for these travelers is generated from origin-destination tables that are independent of the person’s demographics. These itinerant travelers can be viewed as a separate population in equity studies. Every itinerant traveler owns one vehicle. The vehicle is given a unique number and type and is placed in the vehicle file. The starting location of these vehicles is the parking location where the traveler’s trip begins. These starting points are most likely on the boundary of the study area, as itinerant travelers are those that are passing through the area or entering the area from the outside.

1.2.4 Enhancements and Future Developments

Many enhancements could be added to the Population Synthesizer. If a methodology exists to forecast the STF-3A data, then this data could be used to produce a forecast synthetic population. With a slight modification of the code, one would not need to use the variables listed above as the inputs. Under the assumption that the household correlation structure does not change over time, the forecast STF-3A data could be used as the marginal counts for a multiway table, and the generation could proceed as above.

Enhancements could also be made to the number and type of vehicles assigned to the synthetic households. Demographics and household location could be used in a vehicle-ownership model, and the number and emissions type of vehicles for each household could be reassigned.

Usually, only one activity location is assigned to each link on the *walk* network. There is nothing to prohibit the addition of more activity locations to the links. This way, the households could be spread out along the street that is represented by the link. The effect of doing this is unknown.

1.2.5 References

Beckman, Richard J., Baggerly, Keith A. and McKay, Michael D. (1996), Creating Synthetic Baseline Populations, *Transportation Research A*, Vol. 30, No. 6, pp 415-429.


Census (1992a) *Census of Population and Housing, 1990*; Summary Tape File 3 on CD-ROM Technical Documentation/prepared by the Bureau of Census. The Bureau, Washington.

Census (1992b) *Census of Population and Housing, 1990*; Public Use Microdata Sample U.S. Technical Documentation/prepared by the Bureau of Census. The Bureau, Washington.


Deming, W. E. and Stephan, F. F. (1940), On A Least Squares Adjustment Of A Sampled Frequency Table When The Expected Marginal Tables Are Known, *Annals of Mathematical Statistics*, Vol. 11, pp 427-444.

1.3 Usage

The Population Synthesizer module contains several programs that are executed in order and prepare files for the subsequent modules in the TRANSIMS process. The tasks to be done within this module are described in the following subsections.

NOTE: The symbol  is used to indicate items that apply to the June release only.

1.3.1 Assemble the Required Input Data for Synthetic Population Generation

 For the June release, sample input data is supplied with the TRANSIMS-LANL software in `$TRANSIMS_HOME/data/synpop`. You may utilize this data or go through the general procedure described here to obtain similar data. Be aware that the population data must correspond with the supplied transportation network data in order to complete subsequent tasks in the TRANSIMS analysis process, so use the general procedure advisedly.

PUMS and STF-3A data is available on CD-ROM from the Census Bureau.

MABLE/GEOCORR data is obtained from the following WWW site:

<http://plue.sedac.ciesin.org/plue/geocorr/>

MABLE/GEOCORR data must be obtained via a WWW browser prior to running the Population Synthesizer. Perform the following steps to generate a MABLE data file:

- 1) Go to the above web site and select the state in which the PUMS areas of interest are.
- 2) Select the source and destination geocode data (Figure 2 and Figure 3).
- 3) For weighting variable, select Population (1900 Census). Do not check the box *Ignore Census Blocks...* (Figure 4).
- 4) Select *Comma Separated Value File* → *Codes and Names* (Figure 5). No additional output options are required.
- 5) Click *Run Request*. The web server will require several minutes to generate the file. When it is finished, a page with a *geocorr.csv* link will be displayed.
- 6) Click on the link to view the MABLE data.
- 7) After the data has downloaded, use your browser's *Save File* feature to save the data to disk. It is advisable to save the file with a meaningful name, such as *NewMexico_MABLE.csv*. The file name extension **must** be *.csv*.

Select "SOURCE" Geocode(s)

Entire Universe [no code]

 State (1990)
 County (1990)
 County Subdivision: MCD (1990)
 Place: City, Town, Village, etc. (1990)
 Census Tract/BNA (1990)
 Census Block Group (1990)
 Census Block (1990)
 5-digit Postal ZIP Code (circa 7/1991)

 Concentric Ring Pseudo-Geocode (*)
 Hydrologic Unit Code: HUC (***)

 Urban-Rural Portion (1990)
 Metro Area: MSA or CMSA (1990)
 Primary MSA: PMSA (1990)
 Urbanized Area: UA (1991)
 Congressional District: 102nd (1990)
 Congressional District: 103rd (1992)
 PUMA: Code A, 5% Sample (PUMS 1990)

Figure 2: Select "SOURCE" Geocode(s) window.

Select "TARGET" Geocode(s)

Entire Universe [no code]

 State (1990)
 County (1990)
 County Subdivision: MCD (1990)
 Place: City, Town, Village, etc. (1990)
 Census Tract/BNA (1990)
 Census Block Group (1990)
 Census Block (1990)
 5-digit Postal ZIP Code (circa 7/1991)

 Concentric Ring Pseudo-Geocode (*)
 Hydrologic Unit Code: HUC (***)

 Urban-Rural Portion (1990)
 Metro Area: MSA or CMSA (1990)
 Primary MSA: PMSA (1990)
 Urbanized Area: UA (1991)
 Congressional District: 102nd (1990)
 Congressional District: 103rd (1992)
 PUMA: Code A, 5% Sample (PUMS 1990)

Figure 3: Select "TARGET" Geocode(s) window.

Weighting Variable:

Specify the weighting variable to use for determining the portion of the source geocodes corresponding to the target geocodes:

☐ Population (1990 census)
☐ Land Area (square km)
☐ Housing Units (1990 census)

☐ Ignore Census Blocks with a value of 0 for the weighting variable.

Figure 4: Select Weighting Variable.

Comma Separated Value File

☐ Generate a CSV file

☐ Just Codes (No Names)
☒ Codes and Names
☐ Just Names (No Codes)

☐ Use tabs (not commas) as delimiter

• Process time for large areas may be several minutes.

Figure 5: Comma-Separated Value File window.

1.3.2 Run the Population Synthesizer

Before the Population Synthesizer can be run, three user environment variables must be set.

The environment variables, with sample values are shown here (Bourne shell).

```
export STF_INFO_DIR=$TRANSIMS_HOME/data/synpop/Parep2/stf
export STF_DATA_DIR=/cdrom
export PUMS_DATA_DIR=/cdrom
```

✎ export STF_DATA_DIR=\$TRANSIMS_HOME/data/synpop/stf
export PUMS_DATA_DIR=\$TRANSIMS_HOME/data/synpop/pums

In the C-shell environment, the variables are set with setenv:

```
setenv STF_INFO_DIR $TRANSIMS_HOME/data/synpop/Parep2/stf
setenv STF_DATA_DIR /cdrom
setenv PUMS_DATA_DIR /cdrom
```

```

setenv STF_DATA_DIR $TRANSIMS_HOME/data/synpop/stf
setenv PUMS_DATA_DIR $TRANSIMS_HOME/data/synpop/pums

```

Refer to Section 1.4 for a step-by-step tutorial on the usage of the Population Synthesizer. Three synthetic household output files are produced by this program.

The iterative proportional fitting computational process requires a fairly large amount of memory. The Population Synthesizer was run on PUMAS 01000, 01200, 01300, 01400, and 01500 for the urban area 6442 (Portland--Vancouver, OR--WA (pt.): FIPS.STATE=41, URBAREA=6442), and the running process required 110.6 MB of memory at its peak usage. For this reason, it is advisable to run one PUMA at a time for areas of interest.

The synthetic population output files are large as well. The family synthetic population file for the above-mentioned case was 67.4 MB, the non-family file was 13.2 MB, and the group quarters file was 0.9 MB. The processing time for this set of PUMAs was approximately four hours from start to finish, again on a 266 MHz Pentium machine running Linux.

1.3.3 Locate a Population on a Transportation Network

BlockGroupLoc is the program that generates home locations for populations on a transportation network by correlating census tract and block group user data values specified in the network activity location file with tract and block group data in the baseline population. The candidate home locations must have the same census tract and block group as the household and have residential land use values greater than zero. *BlockGroupLoc* also generates the household and person IDs and assigns them to the located population. The user data in the activity location table in the TRANSIMS transportation network must contain tract, block group, and residential and commercial land use values.

Usage: *BlockGroupLoc* <configuration file>

BlockGroupLoc uses the following configuration file keys from the TRANSIMS configuration file. Some keys have default values that may be used if the key is not specified in the configuration file.

Table 9: *BlockGroupLoc* configuration keys.

Configuration Key	Description
POP_BASELINE_FILE	The name of the file containing the baseline population.
POP_LOCATED_FILE	The name of the file where the located population will be written.
ACT_HOME_HEADER	The user data column header in the network activity location file used to specify single family home locations. Default = <i>HOME</i> .
ACT_MULTI_FAMILY_HEADER	The user data column header in the network activity location file used to specify multifamily home locations. If not specified, multifamily user data from the activity location file is ignored.
ACT_TRACT_HEADER	The user data column header in the network activity location file used to specify the census tract. Default = <i>TRACT</i> .
ACT_BLOCKGROUP_HEADER	The user data column header in the network activity location file used to specify the block group. Default = <i>BG</i> .
POP_STARTING_HH_ID	The number from which the generated households will be sequentially numbered. Default = 1.

Configuration Key	Description
POP_STARTING_PERSON_ID	The number from which the generated persons will be sequentially numbered. Default = 101.
NET_DIRECTORY	The directory where the network files reside.
NET_NODE_TABLE	The network node table name.
NET_LINK_TABLE	The network link table name.
NET_ACTIVITY_LOCATION_TABLE	The network activity location table name.

☛ The current version of *BlockGroupLoc* must be executed once for each of the three types of synthetic household files produced by running the Population Synthesizer. Specify distinct values for POP_BASELINE_FILE and POP_LOCATED_FILE in each run. Also, specify different values for POP_STARTING_HH_ID and POP_STARTING_PERSON_ID in each run such that IDs in the output files do not overlap. It may be necessary to examine the successive output files to determine that last value of each ID used thus far. Specify the values Sfr-area for ACT_HOME_HEADER, Mfr-area for ACT_MULTI_FAMILY_HEADER, and BLOCKGR for ACT_BLOCKGROUP_HEADER.

☛ Following the execution of *BlockGroupLoc* to produce three output files, these three files must be concatenated into a single file. Use a text editor to remove the two header lines from the second and third files, and append each file to the first file. The resulting concatenated file will be used as an input file for the *Vehgen* program described in the following section and for the NISS Activity Generator. There will be fewer records in the located population than were in the household files because of the limited size of the network.

1.3.4 Generate a TRANSIMS Vehicle File for Located Synthetic Populations

A TRANSIMS vehicle file contains information about the initial locations of a household's vehicles. For most households, the starting location of the vehicle will be the parking location that is near the household's home location.

Vehgen is the program that creates a TRANSIMS vehicle file that contains information about the household's vehicles and their starting locations. Refer to Volume 3—*Files*, Section 4.2, for a description of the format of a TRANSIMS vehicle file. Each vehicle's starting parking location is found by iterating through the process links that are connected to the home activity location. Every home activity location must have at least one parking location that is accessible via the activity location's process links.

Usage: Vehgen <configuration file>




Vehgen uses the following configuration file keys from the TRANSIMS configuration file. Some keys have default values which may be used if the key is not specified in the configuration file.

Table 10: Vehgen configuration keys.

Configuration Key	Description
POP_LOCATED_FILE	The name of the file containing the located population.
VEHICLE_FILE	The name of the TRANSIMS vehicle file that will be written.
POP_STARTING_VEHICLE_ID	The number from which the vehicle IDs will be sequentially numbered. Default = 100.
NET_DIRECTORY	The directory where the network files reside.
NET_NODE_TABLE	The network node table name.
NET_LINK_TABLE	The network link table name.
NET_ACTIVITY_LOCATION_TABLE	The network activity location table name.
NET_PARKING_TABLE	The network parking table name.
NET_TRANSIT_STOP_TABLE	The network transit stop table name.
NET_PROCESS_LINK_TABLE	The network process link table name.

1.4 Tutorial

Perform the following steps to run the Population Synthesizer. The environmental variables in Section 1.3.2 must be set before running the Population Synthesizer.

- 1) Create a working directory and copy the MABLE/GEOCORR .csv file into that directory.
 If data is being read from \$TRANSIMS_HOME/data/synpop, copy the file *NewMexico_MABLE.csv* from there to the working directory.
- 2) Change directory (cd) into the working directory.
- 3) Start the Population Synthesizer by typing the executable name, *Syn*. Note that the directory where *Syn* is installed must be in the user's path (*TRANSIMS_HOME/bin/Syn*).
- 4) Synthetic populations are generated by the following sequence of operations (Figure 6):
 - a) Type a two-letter state abbreviation.
 If data is being read from \$TRANSIMS_HOME/data/synpop, enter nm (in lower case).
 - b) Enter random number seeds, if desired. Up to three integer values may be used as random seeds. If no data is entered, zeros will be used as the random number seeds.
 - c) Type the PUMS ID. This is the ID of the PUMS area for which synthetic populations are to be generated. You must press **[Enter]** after the PUMS ID.
 If data is being read from \$TRANSIMS_HOME/data/synpop, enter 00400 and press **[Enter]**.
 - d) Type the base output file name. This should be a meaningful string, and it will be prepended to the output synthetic population file names.
 - e) Extract the PUMS data for the selected PUMS IDs. This is done by clicking *[Press For Options]* in the upper left corner of the Population Synthesizer window and selecting the first menu item, *Extract PUMS Data*. A warning pop-up window with the message “If PUMS data is on cdrom, mount it now.” will be displayed. Click *[OK]* after the CD is mounted.

☛ If data is being read from *\$TRANSIMS_HOME/data/synpop*, click [OK].

An input pop-up window with the message “*PUMS directory name?*” and a default value equivalent to the value of *PUMS_DATA_DIR* will be displayed. If this is the correct source of the PUMS data, simply click [OK]. Otherwise, modify the directory name as necessary, and then click [OK].

- f) At this time, household and person demographics may be specified. Click [*Press For Options*] and select either “*Specify Household Demographics*” or “*Specify Person Demographics*.” A window containing the names of demographic data will be displayed. These are the demographics as they are defined in the *pumsusdd.txt* data dictionary file that is on the PUMS CD-ROM. Each of the demographic button names is defined in the *pumsusdd.txt* data file (Section 1.4.1) supplied by the U. S. Census Bureau with their 1990 U. S. Public Use Microdata Sample (PUMS) 5% data. Click on each button for which you wish to have demographic data saved in the output synthetic population files. Be sure to click [*Save*] on each page of demographic data names specification. The demographic data is saved to disk, so this data need only be entered once, or until a different set of demographics is required.

Several household demographics are required for TRANSIMS and are automatically included whether or not they are selected by the user. These include the PUMSHH (household id), PERSONS (number of persons), and AUTOS (number of vehicles available).

☛ Several additional demographics are required by the NISS Activity Generator (Section 2). Include R18UNDR, RWRKR89, and RHHINC in your selections for household demographics. Also include AGE, RELAT1, SEX, and WORK89 in your selections for person demographics.

- g) Click [*Press For Options*] and select the “*Select MABLE CSV File, Generate Populations*” option. A file selector window will appear in which the names of all MABLE/GEOCORR .csv files are listed. Double click on the appropriate file name.

☛ If data is being read from *\$TRANSIMS_HOME/data/synpop*, select the *NewMexico_MABLE.csv* file that you copied into the current directory in step 1.

After a few seconds, a pop-up window with the message “*If STF3A data is on CDROM, mount the CD now.*” will be displayed. Mount the CD-ROM containing the STF3A data, then click [OK].

☛ If the data is being read from *\$TRANSIMS_HOME/data/synpop*, click [OK].

Syn will now read input data (PUMS and MABLE/GEOCORR data from the current working directory, as well as STF3A data); and it will generate several intermediate files that will be used as inputs to the iterative proportional fitting (rake) section of *Syn*. The status indicators (the text sub-window above the “State” input box, and the “Percent Done” dial indicator) provide the user with information on processing progress. When processing large PUMAs, it may take as long as 5-10 minutes per PUMA to complete this phase of the task on a 266 MHz Pentium machine running Linux 2.0. The final processing phase is to generate the population files.

When the status window contains the message “Done Generating Group Quarters Population”, the last of the three synthetic population files has been generated, and the run is completed. The file names are:

- *user_supplied_string_Family_Synthetic_HHRecs.out*
- *user_supplied_string_Non_Family_Synthetic_HHRecs.out*
- *user_supplied_string_Group_Synthetic_HHRecs.out*

The format of these files is two lines containing the names of the household and person demographic data that was collected, followed by the synthetic household data.

The first line of data will consist of the

- tract id,
- block group id,
- an “H” to indicate that it is a household record,
- household id, (always -1)
- PERSONS field,
- AUTOS field,
- home location, (always -1)
- demographic data in the order listed in line one of the file.

This household record is followed by person records (one per person in the household files; group quarter records have one person per household).

A person record consists of the

- household id, (always -1)
- a “P” to indicate that it is a person record,
- person id, (always -1)
- demographic data in the order listed in line two of the file.

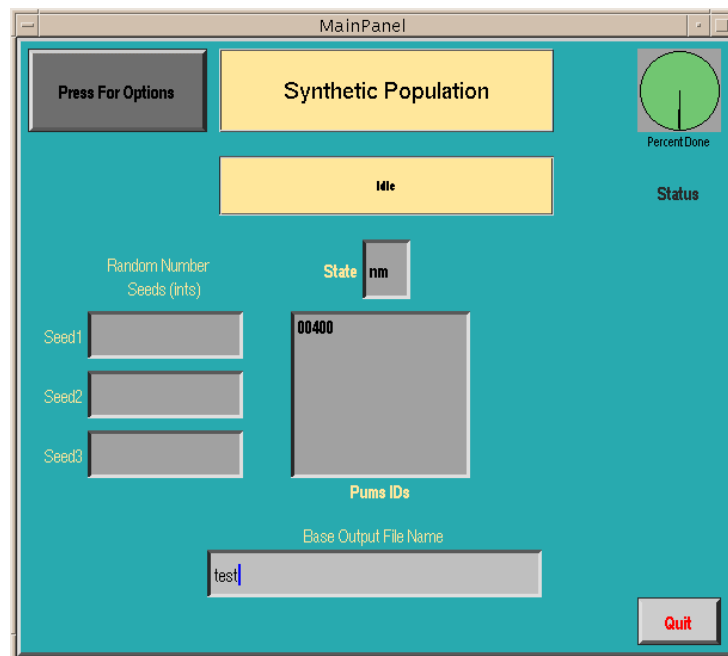


Figure 6: Syn MainPanel.

1.4.1 Demographic Button Name Definitions

Following is a list of the demographic options that are available for households and the allowed values for each demographic item.

DATA	SIZE	BEGIN
D RECTYPE	1	1
Record Type		
V H		.Housing Record
D SERIALNO	7	2
V 0000000..		
9999999		.Housing unit/GQ person serial number unique .identifier assigned within state or state group
D SAMPLE	1	9
Sample Identifier		
V 1		.5% sample
V 2		.1% sample
V 3		.Elderly
D DIVISION	1	10
Division code		
V 0		.Region/division not identifiable
V		.(Selected MSA/PMSAs on 1% sample)
V 1		.New England (Northeast region)
V 2		.Middle Atlantic (Northeast region)
V 3		.East North Central (Midwest region)
V 4		.West North Central (Midwest region)
V 5		.South Atlantic (South region)
V 6		.East South Central (South region)
V 7		.West South Central (South Region)
V 8		.Mountain (West region)
V 9		.Pacific (West region)
D STATE	2	11
State Code		
V 01..56		.FIPS state code (See appendix I-59)
V 99		.PUMA boundaries cross state lines - 1% file
D PUMA	5	13
Public use microdata area (state dependent)		
V 00100..		
V 99999		.PUMA code (Includes tract groups) 1st 3
V		.Digits = main PUMA - generally county place
V		.Last 2 digits = groups of tracts, BNA, etc.
D AREATYPE	2	18
Area type revised for PUMS equivalency file (See Appendix C-1)		
V 10		.Central city
V 11		.Central city part
V 20		.MSA/PMSA - Outside central city
V 21		.MSA/PMSA - Outside central city (part)
V 22		.Central City (part) & outside central city
V		.(part)
V 30		.Entire MSA
V 31		.2 or more MSAs/PMSAs
V 40		.Mixed MSA/PMSA/NON-MSA/PMSA area
V 50		.Outside MSA/PMSA
V 60		.Place


```

V      61 .Place - part
V      70 .MCDs/Towns (New England only)
V      80 .Counties/independent Cities (2 or more)
V      81 .County/independent city - part
V      82 .County/independent city

D  MSAPMSA      4      20
    MSA/PMSA
V    0040..
V    9360 .FIPS/MSA/PMSA code, selected MSA/PMSA
V        .(See appendix G)
V    9997 .Mixed MSA/PMSA NONMSA/PMSA area
V    9998 .2 or more MSAs
V    9999 .Not in MA

D  PSA          3      24
    Planning service area (elderly sample only -
    state dependent)
V    000 .N/A (Elderly PUMS only)
V  1..18B .Planning service area codes (See appendix G)

D  SUBSAMPL     2      27
    Subsample number (Use to pull extracts - 1/1000/etc.)
V    00..99 .See text. pp 4-45.

D  HOUSWGT      4      29
    Housing Weight
V    0000..
V    1152 .Integer weight of housing unit

D  PERSONS      2      33
    Number of person records following this housing
    record
V    00 .Vacant unit
V    01 .One person record (one person in household
V        .or any person in group quarters)
V    02..29 .Number of person records (number of persons
V        .in household)

D  GQINST       1      35
    Group quarters institution
V    0 .N/A (housing unit)
V    1 .Institutionalized
V    2 .Not institutionalized

D  HFILLER      3      36
    Filler

D  UNITS1       2      39
    Units in structure
V    00 .N/A (GQ)
V    01 .Mobile home or trailer
V    02 .One-family house detached
V    03 .One-family house attached
V    04 .2 Apartments
V    05 .3-4 Apartments
V    06 .5-9 Apartments
V    07 .10-19 Apartments
V    08 .20-49 Apartments
V    09 .50 or more apartments
V    10 .Other

D  HUSFLAG      1      41
    All 100% housing unit data substituted
V    0 .No
V    1 .Yes

```

```

D  PDSFLAG          1          42
    All 100% person data substituted
V          0  .No
V          1  .Yes

D  ROOMS            1          43
    Rooms
V          0  .N/A (GQ)
V          1  .1 Room
V          2  .2 Rooms
V          3  .3 Rooms
V          4  .4 Rooms
V          5  .5 Rooms
V          6  .6 Rooms
V          7  .7 Rooms
V          8  .8 Rooms
V          9  .9 or more rooms

D  TENURE            1          44
    Tenure
V          0  .N/A (GQ/vacant)
V          1  .Owned with mortgage or loan
V          2  .Owned free and clear
V          3  .Rented for cash rent
V          4  .No cash rent

D  ACRE10            1          45
    On ten acres or more
V          0  .N/A (GQ/not a one-family house or mobile home)
V          1  .House on ten or more acres
V          2  .House on less than ten acres

D  COMMUSE           1          46
    Business or medical office on property
V          0  .N/A (GQ/not a one-family house or mobile home)
V          1  .Yes
V          2  .No

D  VALUE             2          47
    Property value
V          00  .N/A (GQ/rental unit/vacant, not for sale only)
V          01  .Less than $ 10000
V          02  . $ 10000 - $ 14999
V          03  . $ 15000 - $ 19999
V          04  . $ 20000 - $ 24999
V          05  . $ 25000 - $ 29999
V          06  . $ 30000 - $ 34999
V          07  . $ 35000 - $ 39999
V          08  . $ 40000 - $ 44999
V          09  . $ 45000 - $ 49999
V          10  . $ 50000 - $ 54999
V          11  . $ 55000 - $ 59999
V          12  . $ 60000 - $ 64999
V          13  . $ 65000 - $ 69999
V          14  . $ 70000 - $ 74999
V          15  . $ 75000 - $ 79999
V          16  . $ 80000 - $ 89999
V          17  . $ 90000 - $ 99999
V          18  . $100000 - $124999
V          19  . $125000 - $149999
V          20  . $150000 - $174999
V          21  . $175000 - $199999
V          22  . $200000 - $249999
V          23  . $250000 - $299999
V          24  . $300000 - $399999
V          25  . $400000 or more

```

D	RENT1	2	49
	Monthly rent		
V	00	.N/A (GQ/not a rental unit)	
V	01	.Less than \$ 80	
V	02	.\$ 80 - \$ 99	
V	03	.\$ 100 - \$124	
V	04	.\$ 125 - \$149	
V	05	.\$ 150 - \$174	
V	06	.\$ 175 - \$199	
V	07	.\$ 200 - \$224	
V	08	.\$ 225 - \$249	
V	09	.\$ 250 - \$274	
V	10	.\$ 275 - \$299	
V	11	.\$ 300 - \$324	
V	12	.\$ 325 - \$349	
V	13	.\$ 350 - \$374	
V	14	.\$ 375 - \$399	
V	15	.\$ 400 - \$424	
V	16	.\$ 425 - \$449	
V	17	.\$ 450 - \$474	
V	18	.\$ 475 - \$499	
V	19	.\$ 500 - \$524	
V	20	.\$ 525 - \$549	
V	21	.\$ 550 - \$599	
V	22	.\$ 600 - \$649	
V	23	.\$ 650 - \$699	
V	24	.\$ 700 - \$749	
V	25	.\$ 750 - \$999	
V	26	.\$1000 or more	
V	27	.No cash rent (NCR)	

D	MEALS	1	51
	Meals included in rent		
V	0	.N/A (GQ/not a rental unit/rental-NCR)	
V	1	.Yes	
V	2	.No	

D	VACANCY1	1	52
	Vacant usual home elsewhere (UHE)		
V	0	.N/A (occupied or regular vacant/GQ)	
V	1	.Vacant UHE-owner	
V	2	.Vacant UHE-renter	
V	3	.Vacant UHE-undetermined	

D	VACANCY2	1	53
	Vacancy status		
V	0	.N/A (occupied/GQ)	
V	1	.For rent	
V	2	.For sale only	
V	3	.Rented or sold, not occupied	
V	4	.For seasonal/recreational/occasional use	
V	5	.For migratory workers	
V	6	.Other vacant	

D	VACANCY3	1	54
	Boarded up status		
V	0	.N/A (occupied/GQ)	
V	1	.Yes	
V	2	.No	

D	VACANCY4	1	55
	Months vacant		
V	0	.N/A (occupied/GQ)	
V	1	.Less than 1 month	
V	2	.1 up to 2 months	
V	3	.2 up to 6 months	
V	4	.6 up to 12 months	

V	5	.12 up to 24 months	
V	6	.24 or more months	
D	YRMOVED	1	56
		When moved into this house or apartment	
V	0	.N/A (GQ/vacant)	
V	1	.1989 or 1990	
V	2	.1985 to 1988	
V	3	.1980 to 1984	
V	4	.1970 to 1979	
V	5	.1960 to 1969	
V	6	.1959 or earlier	
D	BEDROOMS	1	57
		Bedrooms	
V	0	.N/A (GQ)	
V	1	.No bedrooms	
V	2	.1 Bedroom	
V	3	.2 Bedrooms	
V	4	.3 Bedrooms	
V	5	.4 Bedrooms	
V	6	.5 or more bedrooms	
D	PLUMBING	1	58
		Complete plumbing facilities	
V	0	.N/A (GQ)	
V	1	.Yes, all three facilities	
V	2	.No	
D	KITCHEN	1	59
		Complete kitchen facilities	
V	0	.N/A (GQ)	
V	1	.Yes	
V	2	.No	
D	TELEPHON	1	60
		Telephone in Unit	
V	0	.N/A (GQ/vacant)	
V	1	.Yes	
V	2	.No	
D	AUTOS	1	61
		Vehicles (1 ton or less) available	
V	0	.N/A (GQ/vacant)	
V	1	.No vehicles	
V	2	.1 vehicle	
V	3	.2 vehicles	
V	4	.3 vehicles	
V	5	.4 Vehicles	
V	6	.5 Vehicles	
V	7	.6 Vehicles	
V	8	.7 or more vehicles	
D	FUELHEAT	1	62
		House heating fuel	
V	0	.N/A (GQ/vacant)	
V	1	.Gas: Underground pipes	
V	2	.Gas: Bottled, tank, or LP	
V	3	.Electricity	
V	4	.Fuel oil, kerosene, etc.	
V	5	.Coal or coke	
V	6	.Wood	
V	7	.Solar energy	
V	8	.Other fuel	
V	9	.No fuel used	
D	WATER	1	63

Source of water
V 0 .N/A (GQ)
V 1 .Public system or private company
V 2 .Individual drilled well
V 3 .Individual dug well
V 4 .Other source such as a spring, creek, etc.

D SEWAGE 1 64
Sewage disposal
V 0 .N/A (GQ)
V 1 .Public sewer
V 2 .Septic tank or cesspool
V 3 .Other means

D YRBUILT 1 65
When structure first built
V 0 .N/A (GQ)
V 1 .1989 or 1990
V 2 .1985 to 1988
V 3 .1980 to 1984
V 4 .1970 to 1979
V 5 .1960 to 1969
V 6 .1950 to 1959
V 7 .1940 to 1949
V 8 .1939 or earlier

D CONDO 1 66
House or apartment part of condominium
V 0 .N/A (GQ)
V 1 .Yes
V 2 .No

D ONEACRE 1 67
House on less than 1 acre
V 0 .N/A (GQ, two or more units in structure)
V 1 .Yes
V 2 .No

D AGSALES 1 68
1989 Sales of Agriculture Products
V 0 .N/A (less than 1 acre/GQ/vacant/
V .2 or more units in structure)
V 1 .None
V 2 . \$1 to \$999
V 3 . \$1,000 to \$2,499
V 4 . \$2,500 to \$4,999
V 5 . \$5,000 to \$9,999
V 6 . \$10,000 or more

D ELECCOST 4 69
Electricity (yearly cost)*
V 0000 .N/A (GQ/vacant)
V 0001 .Included in rent or in condo fee
V 0002 .No charge or electricity not used
V 0003..
3099 . \$3 to \$3,099
V 3100 .Topcode
V 3101+ . \$3101 or more = state median of topcoded
.values

D GASCOST 4 73
Gas (yearly cost)*
V 0000 .N/A (GQ/vacant)
V 0001 .Included in rent or in condo fee
V 0002 .No charge or gas not used
V 0003..
2099 . \$3 to \$2,099

```

V      2100 .Topcode
V      2101+ . $2101 or more = state median of topcoded
           .values

D  WATRCOST      4      77
      Water (yearly cost)
V      000 .N/A (GQ/vacant)
V      001 .Included in rent or in condo fee
V      002 .No charge
V  003..999 . $3 to $999
V      1000 .Topcode
V      1000+ . $1001+ or more = state median of topcoded
           .values

D  FUELCOST      4      81
      House heating fuel (yearly cost)
V      0000 .N/A (GQ/vacant)
V      0001 .Included in rent or in condo fee
V      0002 .No charge or these fuels not used
V      0003..
      1899 . $3 to $1,899
V      1900 .Topcode
V      1,901+ . $1,901 or more = state median of topcoded
           .value

D  RTAXAMT      2      85
      Property taxes (yearly amount)
V      00 .N/A (GQ/vacant/not owned or being bought/not a
           .one-family house, mobile home or trailer or
           .condo)
V      01 .None
V      02 . $ 2 - $ 49
V      03 . $ 50 - $ 99
V      04 . $ 100 - $ 149
V      05 . $ 150 - $ 199
V      06 . $ 200 - $ 249
V      07 . $ 250 - $ 299
V      08 . $ 300 - $ 349
V      09 . $ 350 - $ 399
V      10 . $ 400 - $ 449
V      11 . $ 450 - $ 499
V      12 . $ 500 - $ 549
V      13 . $ 550 - $ 599
V      14 . $ 600 - $ 649
V      15 . $ 650 - $ 699
V      16 . $ 700 - $ 749
V      17 . $ 750 - $ 799
V      18 . $ 800 - $ 849
V      19 . $ 850 - $ 899
V      20 . $ 900 - $ 949
V      21 . $ 950 - $ 999
V      22 . $1000 - $1099
V      23 . $1100 - $1199
V      24 . $1200 - $1299
V      25 . $1300 - $1399
V      26 . $1400 - $1499
V      27 . $1500 - $1599
V      28 . $1600 - $1699
V      29 . $1700 - $1799
V      30 . $1800 - $1899
V      31 . $1900 - $1999
V      32 . $2000 - $2099
V      33 . $2100 - $2199
V      34 . $2200 - $2299
V      35 . $2300 - $2399
V      36 . $2400 - $2499
V      37 . $2500 - $2599

```

```

V      38  .$.2600 - $.2699
V      39  .$.2700 - $.2799
V      40  .$.2800 - $.2899
V      41  .$.2900 - $.2999
V      42  .$.3000 - $.3099
V      43  .$.3100 - $.3199
V      44  .$.3200 - $.3299
V      45  .$.3300 - $.3399
V      46  .$.3400 - $.3499
V      47  .$.3500 - $.3599
V      48  .$.3600 - $.3699
V      49  .$.3700 - $.3799
V      50  .$.3800 - $.3899
V      51  .$.3900 - $.3999
V      52  .$.4000 - $.4099
V      53  .$.4100 - $.4199
V      54  .$.4200 - $.4299
V      55  .$.4300 - $.4399
V      56  .$.4400 - $.4499
V      57  .$.4500 = Topcode
V      58  .$.4501 - $.54992Ä;RANGE
V      59  .$.5500 - $.7499 3 FOR
V      60  .$.7500 or more2ÄÜ MEDIAN

D  HFILLER2      3      87

D  INSAMT      4      90
    Fire/hazard/flood insurance (yearly amount)
V      0000  .N/A (not owned or being bought/not a one
V            .family house, mobile home, or condo/GQ/vacant)
V      0001  .None
V      0002..
V            1299  .$.2 to $1,299
V            1300  .Topcode
V            1301+ .$.1,301 or more=state median of topcoded values

D  MORTGAG      1      94
    Mortgage status
V      0      .N/A (not owned or being bought/not a one family
V            .house, mobile home, or condo/GQ/vacant)
V      1      .Mortgage deed of trust, or similar debt
V      2      .Contract to purchase
V      3      .None

D  MORTGAG3      5      95
    Mortgage payment (monthly amount)
V      00000  .N/A (not owned or being bought/not a one
V            .family house, mobile home, or condo/GQ/vacant)
V      00001  .No regular payment required
V      00002..
V            01999 .$.2 to $1,999
V            02000 .Topcode
V      02001+ . $.2,001 or more = state median of topcoded
V            .values

D  TAXINCL      1      100
    Payment include real estate taxes
V      0      .N/A (GQ/vacant/not owned or being bought/
V            .not a one family house or condo/not mortgaged/
V            .No regular mortgage payment)
V      1      .Yes, taxes included in payment
V      2      .No, taxes paid separately or taxes not required

D  INSINCL      1      101
    Payment include fire/hazard/flood insurance
V      0      .N/A (GQ/vacant/not owned or being bought/
V            .Not a one family house, MHT or condo/not

```

```

        .mortgaged/no regular mortgage payment)
V      1 .Yes, insurance included in payment
V      2 .No, insurance paid separately or no insurance

D MORTGAG2      1      102
    Second mortgage or home equity loan status
V      0 .N/A (GQ/vacant/not owned or being bought/
        .not a one family house, mobile home, trailer or
        .condo/not mortgaged/no second mortgage)
V      1 .Yes
V      2 .No

D MORTAMT2      5      103
    Second mortgage payment (monthly amount)
V      00000 .N/A (GQ/vacant/condo/not owned or being
        .bought/not a one family house/not mortgaged/
        .no second mortgage)
V      00001 .No regular payment required
V      00002..
        00999 . $2 to $999
V      01000 .Topcode
V      01001+ . $1001 or more = state median of topcoded
        .values

D CONDOFEE      4      108
    Condo fee (monthly amount)
V      0000 .N/A (not owned or being bought/not
        .condo/GQ/vacant/no condo fee)
V      0001..
        0599 . $1 - $599
V      0600 .Topcode
V      0601+ . $601 or more = state median of topcoded values

D MOBLHOME      4      112
    Mobile home costs (yearly amount)
V      0000 .N/A (GQ/vacant/not owned or being bought/
        .not mobile home/no costs)
V      0001..
        3399 . $1 - $3,399 (cost in dollars)
V      3400 .Topcode
V      3401+ . $3401 or more = state median of topcoded
        .values

D RFARM      1      116
    Farm/nonfarm status
V      0 .N/A (GQ/urban)
V      1 .Rural farm
V      2 .Rural nonfarm

D RGRENT      4      117
    Gross rent
V      0000 .N/A (GQ/vacant, not rented for cash rent)
V      0001..
        1499 .Gross rent (dollars)
V      1500 .Topcode
V      1501+ .1501 or more = state median of topcoded values

D RGRAPI      2      121
    Gross rent as a percentage of household income in
    1989
V      00 .N/A (GQ/vacant/not rented for cash rent/owner
        .occupied/no household income)
V      01 . 1% to 9%
V      02 .10% to 14%
V      03 .15% to 19%
V      04 .20% to 24%
V      05 .25% to 29%

```


V 06 .30% to 34%
 V 07 .35% to 39%
 V 08 .40% to 49%
 V 09 .50% to 59%
 V 10 .60% to 69%
 V 11 .70% to 79%
 V 12 .80% to 89%
 V 13 .90% to 99%
 V 14 .100% or more

D HFILLER3 1 123
 Filler

D ROWNRCST 5 124
 Selected monthly owner costs

V 00000 .N/A (not owned or being bought/not a one
 V .family house, mobile home, or
 V .condo/GQ/vacant/no costs)
 V 00001..
 20299 .Monthly owner costs in dollars
 V 20300 .Topcode

D RNSMOCPI 3 129
 Selected monthly owner costs as a percentage of
 household income in 1989

V 000 .N/A (not owned or being bought/not a one family
 .house, mobile home, or condo/GQ/vacant/no HH
 .income)
 V 001..100 .1% to 100%
 V 101 .101% or more

D RRENTUNT 1 132
 Specified rent unit

V 0 .Not specified rent unit
 V 1 .Specified rent unit

D RVALUNT 1 133
 Specified value unit

V 0 .Not specified value unit
 V 1 .Specified value unit

D RFAMINC 7 134
 Family income

V 0000000 .N/A(GQ/vacant/no income)
 V -999999..
 V 9999999 .Total family income in dollars (See user notes
 .for state maximum and minimum values
 .Includes single person households.)

D RHHINC 7 141
 Household income

V 0000000 .N/A(GQ/vacant/no income)
 V -999999..
 9999999 .Total household income in dollars (See user notes
 .for state maximum and minimum values)

D RWRKR89 1 148
 Workers in family in 1989

V 0 .N/A (GQ/vacant/non-family household)
 V 1 .No workers
 V 2 .1 worker
 V 3 .2 workers
 V 4 .3 or more workers in family

D RHHLANG 1 149
 Household language

V 0 .N/A (GQ/vacant)

```

V          1 .English only
V          2 .Spanish
V          3 .Other Indo-European language
V          4 .Asian or Pacific Island language
V          5 .Other language

D  RLINGISO          1          150
    Linguistic isolation
V          0 .N/A (GQ/vacant)
V          1 .Not linguistically isolated
V          2 .Linguistically isolated

D  RHHFAMTP          2          151
    Household/family type
V          00 .N/A (GQ/vacant)
V          01 .Married-couple family household
V          Other family household:
V          02 .Male householder
V          03 .Female householder
V          Nonfamily household:
V          Male householder:
V          11 .Living alone
V          12 .Not living alone
V          Female householder:
V          21 .Living alone
V          22 .Not living alone

D  RNATADPT          2          153
    Number of own natural born/adopted children in
    household (unweighted)
V          00 .N/A(GQ/vacant/no own natural born/adopted
V          .children)
V          01..28 .Number of own children natural born/adopted
    children in household

D  RSTPCHLD          2          155
    Number of own stepchildren in household (unweighted)
V          00 .N/A(GQ/vacant/no own stepchildren)
V          01..28 .Number of own stepchildren in household

D  RFAMPERS          2          157
    Number of persons in family (unweighted)
V          00 .N/A (GQ/vacant/non-family household)
V          01..29 .Number of persons in family

D  RNRLCHLD*         2          159
    Number of related children in household (unweighted)
V          00 .N/A (GQ/vacant/no related children)
V          01..28 .Number of related children in household

D  RNONREL           1          161
    Presence of nonrelatives in household
V          0 .N/A (No nonrelatives in household/GQ/vacant)
V          1 .1 or more nonrelatives in household

D  R18UNDR           1          162
    Presence of person under 18 years in household
V          0 .N/A (No person under 18 in household/GQ/vacant)
V          1 .1 or more person under 18 in household

D  R60OVER           1          163
    Presence of persons 60 years and over in household
V          0 .N/A (No person 60 and over/GQ/vacant)
V          1 .1 person 60 and over (unweighted)
V          2 .2 or more person 60 and over (unweighted)

D  R65OVER           1          164

```

	Presence of person 65 years and over in household		
V	0	.N/A (No person 65 and over/GQ/vacant)	
V	1	.1 person 65 and over (unweighted)	
V	2	.2 or more person 65 and over (unweighted)	
D	RSUBFAM	1	165
	Presence of subfamilies in Household		
V	0	.N/A (No subfamilies or not	
V		.applicable/GQ/vacant)	
V	1	.1 or more subfamilies	
D	AUNITS1	1	166
	Units in structure allocation		
V	0	.No	
V	1	.Yes	
D	AROOMS	1	167
	Rooms allocation		
V	0	.No	
V	1	.Yes	
D	ATENURE	1	168
	Tenure allocation		
V	0	.No	
V	1	.Yes	
D	AACRES10	1	169
	On ten acres or more allocation		
V	0	.No	
V	1	.Yes	
D	ACOMMUSE	1	170
	Business or medical office on property allocation		
V	0	.No	
V	1	.Yes	
D	AVALUE	1	171
	Value allocation		
V	0	.No	
V	1	.Yes	
D	ARENT1	1	172
	Monthly rent allocation		
V	0	.No	
V	1	.Yes	
D	AMEALS	1	173
	Meals included in rent allocation		
V	0	.No	
V	1	.Yes	
D	AVACNCY2	1	174
	Vacancy status allocation		
V	0	.No	
V	1	.Yes	
D	AVACNCY3	1	175
	Boarded up status allocation		
V	0	.No	
V	1	.Yes	
D	AVACNCY4	1	176
	Months vacant allocation		
V	0	.No	
V	1	.Yes	
D	AYRMOVED	1	177

When moved into this house or apartment allocation

V 0 .No

V 1 .Yes

D ABEDROOM 1 178

 Number of bedrooms allocation

V 0 .No

V 1 .Yes

D APLUMBNG 1 179

 Complete plumbing facilities allocation

V 0 .No

V 1 .Yes

D AKITCHEN 1 180

 Complete kitchen facilities allocation

V 0 .No

V 1 .Yes

D APHONE 1 181

 Telephones in house allocation

V 0 .No

V 1 .Yes

D AVEHICLE 1 182

 Vehicles available by household allocation

V 0 .No

V 1 .Yes

D AFUEL 1 183

 House heating fuel allocation

V 0 .No

V 1 .Yes

D AWATER 1 184

 Source of water allocation

V 0 .No

V 1 .Yes

D ASEWER 1 185

 Sewage disposal allocation

V 0 .No

V 1 .Yes

D AYRBUILT 1 186

 When structure first built allocation

V 0 .No

V 1 .Yes from not answered

V 2 .Yes "don't know"

D ACONDO 1 187

 House or apartment part of condominium allocation

V 0 .No

V 1 .Yes

D AONEACRE 1 188

 House on less than 1 acre allocation

V 0 .No

V 1 .Yes

D AAGSALES 1 189

 1989 Sales of Agricultural Products allocation

V 0 .No

V 1 .Yes

D AELECCST 1 190

 Electricity (yearly cost) allocation

V 0 .No
 V 1 .Yes

D AGASCST 1 191
 Gas (yearly cost) allocation
 V 0 .No
 V 1 .Yes

D AWATRCST 1 192
 Water (yearly cost) allocation
 V 0 .No
 V 1 .Yes

D AFUELCST* 1 193
 House heating fuel (yearly cost) allocation
 V 0 .No
 V 1 .Yes

D ATAXAMT 1 194
 Taxes on property allocation
 V 0 .No
 V 1 .Yes

D AINSAMT 1 195
 Fire, hazard, flood insurance allocation
 V 0 .No
 V 1 .Yes

D AMORTG 1 196
 Mortgage status allocation
 V 0 .No
 V 1 .Yes no answer
 V 2 .Yes from junior mortgage

D AMORTG3 1 197
 Regular mortgage payment allocation
 V 0 .No
 V 1 .Yes

D ATAXINCL 1 198
 Payment include real estate taxes allocation
 V 0 .No
 V 1 .Yes

D AINSINCL 1 199
 Payment include fire, hazard, flood insurance
 allocation
 V 0 .No
 V 1 .Yes

D AMORTG2 1 200
 Second mortgage status allocation
 V 0 .No
 V 1 .Yes

D AMRTAMT2 1 201
 Second mortgage payment allocation
 V 0 .NO
 V 1 .Yes

D ACNDOFEE 1 202
 Condominium fee allocation
 V 0 .No
 V 1 .Yes

D AMOBLHME 1 203

```

        Mobile home costs allocation
V      0 .No
V      1 .Yes

```

```

D  FILLER          28          204

```

Following is a list of the demographic options that are available for persons and the allowed values for each demographic item.

DATA	SIZE	BEGIN
D RECTYPE	1	1
Record Type		
V P	.Person Record	
D SERIALNO	7	2
V 0000000..		
V 9999999	.Housing unit/GQ person serial number unique	
V	.identifier assigned within state or state group	
D RELAT1	2	9
Relationship		
V 00	.Householder	
V 01	.Husband/wife	
V 02	.Son/daughter	
V 03	.Stepson/stepdaughter	
V 04	.Brother/sister	
V 05	.Father/mother	
V 06	.Grandchild	
V 07	.Other relative	
	Not related	
V 08	.Roomer/boarder/foster child	
V 09	.Housemate/roommate	
V 10	.Unmarried partner	
V 11	.Other nonrelative	
	Group quarters	
V 12	.Institutionalized person	
V 13	.Other persons in group quarters	
D SEX	1	11
Sex		
V 0	.Male	
V 1	.Female	
D RACE	3	12
Recoded detailed race code (Appendix C)		
V 001-037	.(See appendix C)	
V 301-327	.American Indian Tribes	
D AGE	2	15
Age		
V 00	.Less than 1 year	
V 01..89	.Age in years	
V 90	.90 or more years old	
D MARITAL	1	17
Marital status		
V 0	.Now married, except separated	
V 1	.Widowed	
V 2	.Divorced	
V 3	.Separated	
V 4	.Never married or under 15 years old	
D PWGT1	4	18
Person's weight		
V 0001..		
V 1152	.Person's weight	

D	PFILLER1	4	22
	Filler		
D	REMPPLPAR	3	26
	Employment status of parents		
V	000	.N/A (not own child of householder, and not .child in subfamily)	
V		Living with two parents:	
V		Both parents in labor force:	
V	111	.Both parents at work 35 or more hours	
V	112	.Father only at work 35 or more hours	
V	113	.Mother only at work 35 or more hours	
V	114	.Neither parent at work 35 or more hours	
V		Father only in labor force:	
V	121	.Father at work 35 or more hours	
V	122	.Father not at work 35 or more hours	
V		Mother only in labor force:	
V	133	.Mother at work 35 or more hours	
V	134	.Mother not at work 35 or more hours	
V	141	Neither parent in labor force	
V		Living with one parent:	
V		Living with father:	
V	211	.Father at work 35 or more hours	
V	212	.Father not at work 35 or more hours	
V	213	.Father not in labor force	
V		Living with mother:	
V	221	.Mother at work 35 or more hours	
V	222	.Mother not at work 35 or more hours	
V	223	.Mother not in labor force	
D	RPOB	2	29
	Place of birth (Recode)		
V	10	.Born in State of residence	
V		Born in other State in the U.S.:	
V	21	.Northeast	
V	22	.Midwest	
V	23	.South	
V	24	.West	
V		U.S. outlying areas:	
V	31	.Puerto Rico	
V	32	.American Samoa	
V	33	.Guam	
V	34	.Northern Marianas	
V	35	.US Virgin Islands	
V	36	.Elsewhere	
V	40	.Born abroad of American parents	
V		Foreign born:	
V	51	.Naturalized citizen	
V	52	.Not a citizen	
D	RSPOUSE	1	31
	Married, spouse present/spouse absent		
V	0	.N/A (less than 15 years old)	
V	1	.Now married, spouse present	
V	2	.Now married, spouse absent	
V	3	.Widowed	
V	4	.Divorced	
V	5	.Separated	
V	6	.Never married	
D	ROWNCHLD	1	32
	*Own child (see Appendix B, page 14)		
V	* 1	.Own child	
V	* 0	.Not own child	
D	RAGECHLD	1	33

```

      Presence and age of own children
V      * 0 .N/A (male)
V      1 .With own children under 6 years only
V      2 .With own children 6 to 17 years only
V      3 .With own children under 6 years and 6 to 17
      .years
V      * 4 .No own children (.incl. females under 16 years)
D RRELCHLD          1          34

      *Related child (see Appendix B, Page 14)
V      * 1 .Related child
V      * 0 .Not related child

D RELAT2           1          35
      Detailed relationship (other relative)
V      0 .N/A (GQ/not other relative)
V      1 .Son-in-law/daughter-in-law
V      2 .Father-in-law/mother-in-law
V      3 .Brother-in-law/sister-in-law
V      4 .Nephew/niece
V      5 .Grandparent
V      6 .Uncle/aunt
V      7 .Cousin
V      8 .Other related by blood or marriage
V      9 .Other relative

D SUBFAM2           1          36
      Subfamily number
V      0 .N/A (GQ/not in a subfamily)
V      1 .In subfamily 1
V      2 .In subfamily 2
V      3 .In subfamily 3

D SUBFAM1           1          37
      Subfamily relationship
V      0 .N/A (GQ/not in a subfamily)
V      1 .Husband/wife
V      2 .Parent in a parent/child subfamily
V      3 .Child in subfamily

D HISPANIC          3          38
      Detailed Hispanic origin code (See appendix I)
V 000,006.. .
      199 . Not hispanic
V 001,210..
      220 .Mexican, mex-am
V 002,261..
      270 .Puerto Rican
V 003,271..
      274 .Cuban
V 221..230 .Central American
V 231..249 .South American
V 275..289 .Dominican
V 004,200..
      209,250..
      260
V 290..401 .Other Hispanic

D POVERTY           3          41
      Person poverty status recode (See appendix B)
V      000 .N/A
V 001..500 .% Below or above poverty status value
V      501 .501% or more of poverty value

D POB               3          44
      Place of birth (Appendix I)
V 001..056 .FIPS State code (U.S. States and D.C.)

```


V 060..099 .Puerto Rico (072) or U.S. outlying area
 V 100..553 .Foreign country
 V 554 .At sea
 V 555 .Abroad, not specified

D CITIZEN 1 47
 Citizenship
 V 0 .Born in the U.S.
 V 1 .Born in Puerto Rico, Guam, and outlying areas
 V 2 .Born abroad of American parents
 V 3 .U.S. citizen by naturalization
 V 4 .Not a citizen of the U.S.

D IMMIGR 2 48
 Year of entry
 V 00 .Born in the U.S.
 V 01 .1987 to 1990
 V 02 .1985 to 1986
 V 03 .1982 to 1984
 V 04 .1980 or 1981
 V 05 .1975 to 1979
 V 06 .1970 to 1974
 V 07 .1965 to 1969
 V 08 .1960 to 1964
 V 09 .1950 to 1959
 V 10 .Before 1950

D SCHOOL 1 50
 School enrollment
 V 0 .N/A (less than 3 years old)
 V 1 .Not attending school
 V 2 .Yes, public school, public college
 V 3 .Yes, private school, private college

D YEARSCH 2 51
 Educational attainment
 V 00 .N/A (less than 3 years old)
 V 01 .No school completed
 V 02 .Nursery school
 V 03 .Kindergarten
 V 04 .1st, 2nd, 3rd, or 4th grade
 V 05 .5th, 6th, 7th, or 8th grade
 V 06 .9th grade
 V 07 .10th grade
 V 08 .11th grade
 V 09 .12th grade, no diploma
 V 10 .High school graduate, diploma or GED
 V 11 .Some college, but no degree
 V 12 .Associate degree in college, occupational program
 V 13 .Associate degree in college, academic program
 V 14 .Bachelor's degree
 V 15 .Master's degree
 V 16 .Professional degree
 V 17 .Doctorate degree

D ANCSTRY1 3 53
 Ancestry - first entry (See appendix I)
 V 001..998 .Ancestry codes - first entry
 V 999 .Not reported

D ANCSTRY2 3 56
 Ancestry - second entry (See appendix I)
 V 000 .No secondary ancestry
 V 001..998 .Ancestry codes
 V 999 .Not reported

D MOBILITY 1 59

Mobility status (lived here on April 1, 1985)
 V 0 .N/A (less than 5 years old)
 V 1 .Yes same house (nonmovers)
 V 2 .No, different house (movers)

D MIGSTATE 2 60
 Migration - State or foreign country code
 (Appendix I)
 V 00 .N/A (person less than 5 years old/lived
 V .in same house in 1985)
 V 01..56 .FIPS state code (U.S. States and D.C.)
 V 72 .Puerto Rico
 V 98 .Other abroad in 1985
 V 99 .State not identified (B sample)

D MIGPUMA 5 62
 Migration PUMA (state dependent)
 V 00000 .N/A (person less than 5 years old/lived in
 .same house in 1985)
 V 00100..
 99800 .Migration PUMA (Not coded to tract level)
 V 99900 .Abroad

D LANG1 1 67
 Language other than English at home
 V 0 .N/A (less than 5 years old)
 V 1 .Yes, speaks another language
 V 2 .No, speaks only English

D LANG2 3 68
 Language spoken at home (See appendix I)
 V 000..600 .N/A (less than 5 years old/speaks only
 .English)
 V 601..999 .Specific language codes

D ENGLISH 1 71
 Ability to speak English
 V 0 .N/A (less than 5 years old/speaks only English)
 V 1 .Very well
 V 2 .Well
 V 3 .Not well
 V 4 .Not at all

D MILITARY 1 72
 Military service
 V 0 .N/A (less than 16 years old)
 V 1 .Yes, now on active duty
 V 2 .Yes, on active duty in past, but not now
 V 3 .Yes, service in reserves or national guard only
 V 4 .No service

D RVETSERV 2 73
 Veteran period of service
 V 00 .N/A (less than 16 years old, no active duty)
 V 01 .September 1980 or later only
 V 02 .May 1975 to August 1980 only
 V 03 .May 1975 to August 1980 and September 1980
 .or later only
 V 04 .Vietnam era, no Korean conflict, no WWII
 V 05 .Vietnam era and Korean conflict, no WWII
 V 06 .Vietnam era and Korean conflict and WWII
 V 07 .February 1955 to July 1964 only
 V 08 .Korean conflict, no Vietnam era, no WWII
 V 09 .Korean conflict and WWII, no Vietnam era
 V 10 .WWII, no Korean conflict, no Vietnam era
 V 11 .Other service

D SEPT80 1 75
Served September 1980 or later
V 0 .(Did not serve this period/less than 16 years
V .old)
V 1 .Served this period

D MAY7580 1 76
Served May 1975 to August 1980
V 0 .(Did not serve this period/less than 16 years
V .old)
V 1 .Served this period

D VIETNAM 1 77
Served Vietnam era (August 1964 - April 1975)
V 0 .(Did not serve this period/less than 16 years
V . old)
V 1 .Served this period

D FEB55 1 78
Served February 1955 - July 1964
V 0 .(Did not serve this period/less than 16 years
V .old)
V 1 .Served this period

D KOREAN 1 79
Served Korean conflict (June 1950 - January 1955)
V 0 .(Did not serve this period/less than 16 years
V .old)
V 1 .Served this period

D WWII 1 80
Served World War II (September 1940 - July 1947)
V 0 .(Did not serve this period/less than 16 years
V .old)
V 1 .Served this period

D PFILLER2 1 81
Filler

D OTHRSERV 1 82
Served any other time
V 0 .(Did not serve this period/less than 16 years
V .old)
V 1 .Served this period

D YRSSERV 2 83
Years of active duty military service
V 00 .N/A (less than 16 years/no active duty military
V .service)
V 01 .1 Year or less of service
V 02..49 .2 to 49 years of service
V 50 .50 or more years of service

D DISABL1 1 85
Work limitation status
V 0 .N/A (less than 16 years, and selected persons in
V .GQs - See appendix C)
V 1 .Yes, limited in kind or amount of work
V 2 .No, not limited

D DISABL2 1 86
Work prevented status
V 0 .N/A(less than 16 years, and selected persons in
V .GQs - See appendix C)
V 1 .Yes, prevented from working
V 2 .No, not prevented from working

D MOBILIM 1 87
Mobility limitation
V 0 .N/A (less than 15 years/institutionalized
V .person, and selected persons in GQs -
V .See appendix C)
V 1 .Yes, has a mobility limitation
V 2 .No, does not have a mobility limitation

D PERSCARE 1 88
Personal care limitation
V 0 .N/A (less than 15 years/institutionalized
V .person, and selected persons in GQs -
V .See appendix C)
V 1 .Yes, has a personal care limitation
V 2 .No, does not have a personal care limitation

D FERTIL 2 89
Number of children ever born
V 00 .N/A (less than 15 years/male)
V 01 .No children
V 02 .1 Child
V 03 .2 Children
V 04 .3 Children
V 05 .4 Children
V 06 .5 Children
V 07 .6 Children
V 08 .7 Children
V 09 .8 Children
V 10 .9 Children
V 11 .10 Children
V 12 .11 Children
V 13 .12 or more children

D RLABOR 1 91
Employment status recode
V 0 .N/A (less than 16 years old)
V 1 .Civilian employed, at work
V 2 .Civilian employed, with a job but not at work
V 3 .Unemployed
V 4 .Armed forces, at work
V 5 .Armed forces, with a job but not at work
V 6 .Not in labor force

D WORKLWK 1 92
Worked last week
V 0 .N/A (less than 16 years old/not at work/
V .unemployed/NILF/Q21A not reported)
V 1 .Worked
V 2 .Did not work

D HOURS 2 93
Hours worked last week
V 00 .N/A (less than 16 years old/not at
V .work/unemployed/NILF)
V 01..98 .1 to 98 hours worked last week
V 99 .99 or more hours worked last week

D POWSTATE 2 95
Place of work - state - (Appendix I)
V 00 .N/A (not a worker--not in the labor force,
V .including persons under 16 years; unemployed;
V .employed, with a job not at work; Armed Forces,
V .With a job but not at work)
V 01-56 .FIPS state code (U.S. States and D.C.)
V 98 .Abroad
V 99 .State not identified

```

D  POWPUMA          5          97
    Place of work PUMA (State dependent)
V   00000 .N/A (not a worker--not in the labor force,
V         .including persons under 16 years;
V         .unemployed; employed, with a job but not at
V         .work; Armed Forces, with a job but not at
V         .work)
V   00100..
V     99800 .PUMA of work (Not coded to tract level)
V     99900 .Abroad

D  MEANS            2          102
    Means of transportation to work
V   00 .N/A (not a worker--not in the labor force,
V     .including persons under 16 years; unemployed;
V     .employed, with a job but not at work; Armed
V     .Forces, with a job but not at work)
V   01 .Car, truck, or van
V   02 .Bus or trolley bus
V   03 .Streetcar or trolley car
V   04 .Subway or elevated
V   05 .Railroad
V   06 .Ferryboat
V   07 .Taxicab
V   08 .Motorcycle
V   09 .Bicycle
V   10 .Walked
V   11 .Worked at home
V   12 .Other method

D  RIDERS           1          104
    Vehicle occupancy
V   0 .N/A (not a worker or worker whose means of
V     .transportation to work was not car, truck,
V     .or van)
V   1 .Drove alone
V   2 .2 People
V   3 .3 People
V   4 .4 People
V   5 .5 People
V   6 .6 People
V   7 .7 to 9 people
V   8 .10 or more people

D  DEPART           4          105
    Time of departure for work - hour and minute
V   0000 .N/A (not a worker or worker who worked at
V         .home)
V   0001..
V     2400 .Time (hour and minute of departure for
V           .work) (2400 midnight)

D  TRAVTIME         2          109
    Travel time to work
V   00 .N/A (not a worker or worker who worked at home)
V   01..98 .1 to 98 minutes to get to work
V   99 .99 Minutes or more to get to work

D  TMPABSNT         1          111
    Temporary absence from work
V   0 .N/A (less than 16 years old/at work/did not
V     .report Q25)
V   1 .Yes, on layoff
V   2 .Yes, on vacation, temporary illness, labor
V     .dispute
V   3 .No

```

D LOOKING 1 112
 Looking for work
 V 0 .N/A (less than 16 years old/at work/did not
 .report Q26A)
 V 1 .Yes
 V 2 .No

D AVAIL 1 113
 Available for work
 V 0 .N/A (less than 16 years/at work/not looking/
 .Q26A = 0/did not report Q26B)
 V 1 .No, already has a job
 V 2 .No, temporarily ill
 V 3 .No, other reasons (in school, etc.)
 V 4 .Yes, could have taken a job

D YEARWRK 1 114
 Year last worked
 V 0 .N/A (less than 16 years old)
 V 1 .1990
 V 2 .1989
 V 3 .1988
 V 4 .1985 to 1987
 V 5 .1980 to 1984
 V 6 .1979 or earlier
 V 7 .Never worked

D INDUSTRY 3 115
 Industry
 V 000 .N/A (less than 16 years old/unemployed who
 .never worked/nlrf who last worked prior to
 .1985)
 V 010..992 .specific industry codes (see appendix I)

D OCCUP 3 118
 Occupation
 V 000 .N/A (less than 16 years old/unemployed who
 .never worked/nlrf who last worked prior to
 .1985)
 V 003..909 .specific occupation codes (see appendix I)

D CLASS 1 121
 Class of worker
 V 0 .N/A (less than 16 years old/unemployed who
 .never worked/NILF who last worked prior to
 .1985)
 V 1 .employee of a private for profit company or
 .business or of an individual, for wages,
 .salary, or commissions
 V 2 .Employee of a private not-for-profit,
 .tax-exempt, or charitable organization
 V 3 .Local government employee (city, county, etc.)
 V 4 .State government employee
 V 5 .Federal government employee
 V 6 .Self-employed in own not incorporated
 .business, professional practice, or farm
 V 7 .Self-employed in own incorporated
 .business, professional practice or farm
 V 8 .Working without pay in family business or farm
 V 9 .Unemployed, last worked in 1984 or earlier

D WORK89 1 122
 Worked last year (1989)
 V 0 .N/A (less than 16 years old)
 V 1 .Worked last year
 V 2 .Did not work last year

D WEEK89 2 123
 Weeks worked last year (1989)
 V 00 .N/A (less than 16 years old/did not work in
 .1989)
 V 01..52 .1 to 52 weeks worked last year

 D HOUR89 2 125
 Usual hours worked per week last year (1989)
 V 00 .N/A (less than 16 years old/did not work in
 .1989)
 V 01..98 .1 To 98 usual hours
 V 99 .99 Or more usual hours

 D REARNING 6 127
 Total person's earnings
 V 000000 .N/A (no earnings)
 V -19996 .Loss of \$19996 or more
 V -19995..
 283999 .Total person's earnings in dollars
 V 284000 .\$284000 = Topcode
 V 284001+ .State medians included

 D RPINCOME 6 133
 Total person's income (signed)
 V 000000 .N/A (no income)
 V -29997 .Loss of \$29997 or more
 V -29996..
 400999 .Total person's income in dollars
 V 401000 .Topcode of total person's income
 V 401001+ .State medians included

 D INCOME1 6 139
 Wages or salary income in 1989
 V 000000 .N/A (less than 16 years old/none)
 V 000001..
 V 139999 .\$.1 - 139,999
 V 140000 .Topcode
 V 140001+ .140001 or more = state median of topcoded
 .values

 D INCOME2 6 145
 Nonfarm self-employment income in 1989 (signed)
 V 000000 .N/A (less than 16 years/none)
 V -09999 .Loss of \$9,999 or more
 V -00001..
 V -09998 .Loss \$1 to \$9,998
 V 000001 .Break even or \$1
 V 000002..
 089999 .\$.2 To \$89999
 V 090000 .Topcode
 V 090001+ .\$.90,001 or more = state median of topcoded
 .values

 D INCOME3 6 151
 Farm self-employment income in 1989 (signed)
 V 000000 .N/A (less than 16 years/none)
 V -09999 .Loss of \$9,999 or more
 V -00001 to
 -09998 .Loss \$1 to \$9,998
 V 1 .Break even or \$1
 V 000002..
 053999 .\$.2 To \$53999
 V 054000 .Topcode
 V 054001+ .\$.54001 or more = state median of
 .topcoded values

 D INCOME4 6 157

```

        Interest, dividends, and net rental income in 1989 (signed)
V      000000 .N/A (less than 15 years/none)
V      -09999 .Loss of $9,999 or more
V      -00001 to
        -09998 .Loss $1 to $9,998
V      1 .Break even or $1
V      000002..
        039999 .$.2 To $39999
V      040000 .Topcode
V      040001+ .$.40001 or more = state median of
        .topcoded values
D      INCOME5      5      163
        Social security income in 1989
V      00000 .N/A (less than 15 years/none)
V      00001..
        16999 .$.1 to $16999
V      17000 .Topcode
V      17001+ .17001 or more = state median of topcoded
        .values

D      INCOME6      5      168
        Public assistance income in 1989
V      00000 .N/A (less than 15 years/none)
V      00001..
        9999 .$.1 To $9999
V      10000 .Topcode
V      10001+ .$.10001 or more = state median

D      INCOME7      5      173
        Retirement income in 1989
V      00000 .N/A (less than 15 years/none)
V      00001..
        29999 .$.1 to $29999
V      30000 .Topcode
V      30001+ .$.30001 or more = state median of topcoded
        .values

D      INCOME8      5      178
        All other income in 1989
V      00000 .N/A (less than 15 years/none)
V      00001..
        19999 .$.1 to $19999
V      20000 .Topcode
V      20001+ .$.20,001 or more = state median of topcoded
        .values

D      AAUGMENT      1      183
        Augmented person (see text pp. C-5)
V      0 .No
V      1 .Yes

D      ARELAT1      1      184
        Relationship allocation flag
V      0 .No
V      1 .Yes

D      ASEX      1      185
        Sex allocation flag
V      0 .No
V      1 .Yes

D      ARACE      1      186
        Detailed race allocation flag
V      0 .No
V      1 .Yes

D      AAGE      1      187

```


	Age allocation flag	
V	0 .No	
V	1 .Yes	
D	AMARITAL 1 188	
	Marital status allocation flag	
V	0 .No	
V	1 .Yes	
D	AHISPAN 1 189	
	Detailed Hispanic origin allocation flag	
V	0 .No	
V	1 .Yes	
D	ABIRTHPL 1 190	
	Place of birth	
V	0 .No	
V	1 .Yes	
D	ACITIZEN 1 191	
	Citizenship allocation flag	
V	0 .No	
V	1 .Yes	
D	AIMMIGR 1 192	
	Year of entry allocation flag	
V	0 .No	
V	1 .Yes	
D	ASCHOOL 1 193	
	School enrollment allocation flag	
V	0 .No	
V	1 .Yes	
D	AYEARSCH 1 194	
	Highest education allocation flag	
V	0 .No	
V	1 .Yes	
D	AANCSTR1 1 195	
	First ancestry allocation flag	
V	0 .No	
V	1 .Yes	
D	AANCSTR2 1 196	
	Second ancestry allocation flag	
V	0 .No	
V	1 .Yes	
D	AMOBLTY 1 197	
	Mobility status allocation flag	
V	0 .No	
V	1 .Yes	
D	AMIGSTATE 1 198	
	Migration state allocation flag	
V	0 .No	
V	1 .Yes	
D	ALANG1 1 199	
	Language other than English allocation flag	
V	0 .No	
V	1 .Yes	
D	ALANG2 1 200	
	Language spoken at home allocation flag	
V	0 .No	
V	1 .Yes	

D	AENGLISH	1	201	
	Ability to speak English allocation flag			
V	0	.No		
V	1	.Yes		
D	AVETS1	1	202	
	Military service allocation flag			
V	0	.No		
V	1	.Yes		
D	ASERVPER	1	203	
	Military periods of service allocation flag			
V	0	.No		
V	1	.Yes		
D	AYRSSERV	1	204	
	Years of military service allocation flag			
V	0	.No		
V	1	.Yes		
D	ADISABL1	1	205	
	Work limitation status allocation flag			
V	0	.No		
V	1	.Yes		
D	ADISABL2	1	206	
	Work prevention status allocation flag			
V	0	.No		
V	1	.Yes		
D	AMOBLLIM	1	207	
	Mobility limitation status allocation flag			
V	0	.No		
V	1	.Yes		
D	APERCARE	1	208	
	Personal care limitation status allocation flag			
V	0	.No		
V	1	.Yes		
D	AFERTIL	1	209	
	Children ever born allocation flag			
V	0	.No		
V	1	.Yes		
D	ALABOR	1	210	
	Employment status recode allocation flag			
V	0	.No		
V	1	.Yes		
D	AHOURS	1	211	
	Hours worked last week allocation flag			
V	0	.No		
V	1	.Yes		
D	APOWST	1	212	
	Place of work state allocation flag			
V	0	.No		
V	1	.Yes		
D	AMEANS	1	213	
	Means of transportation to work allocation flag			
V	0	.No		
V	1	.Yes		
D	ARIDERS	1	214	
	Vehicle occupancy allocation flag			

V	0	.No	
V	1	.Yes	
D	ADEPART	1	215
	Time of departure to work allocation flag		
V	0	.No	
V	1	.Yes	
D	ATRAVTME	1	216
	Travel time to work allocation flag		
V	0	.No	
V	1	.Yes	
D	ALSTWRK	1	217
	Year last worked allocation flag		
V	0	.No	
V	1	.Yes	
D	AINDUSTR	1	218
	Industry allocation flag		
V	0	.No	
V	1	.Yes	
D	AOCCUP	1	219
	Occupation allocation flag		
V	0	.No	
V	1	.Yes	
D	ACCLASS	1	220
	Class of worker allocation flag		
V	0	.No	
V	1	.Yes	
D	AWORK89	1	221
	Worked last year allocation flag		
V	0	.No	
V	1	.Yes	
D	AWKS89	1	222
	Weeks worked in 1989 allocation flag		
V	0	.No	
V	1	.Yes	
D	AHOUR89	1	223
	Usual hours worked per week in 1989 allocation flag		
V	0	.No	
V	1	.Yes	
D	AINCOME1	1	224
	Wages and salary income allocation flag		
V	0	.No	
V	1	.No (derived)	
V	2	.Yes	
D	AINCOME2	1	225
	Nonfarm self-employment income allocation flag		
V	0	.No	
V	1	.No (derived)	
V	2	.Yes	
D	AINCOME3	1	226
	Farm self-employment income allocation flag		
V	0	.No	
V	1	.No (derived)	
V	2	.Yes	
D	AINCOME4	1	227

```

      Interest, dividend, and net rental income allocation flag
V      0 .No
V      1 .No (derived)
V      2 .Yes

D  AINCOME5      1      228
      Social security income allocation flag
V      0 .No
V      1 .No (derived)
V      2 .Yes
D  AINCOME6      1      229
      Public assistance allocation flag
V      0 .No
V      1 .No (derived)
V      2 .Yes

D  AINCOME7      1      230
      Retirement income allocation flag
V      0 .No
V      1 .No (derived)
V      2 .Yes

D  AINCOME8      1      231
      All other income allocation flag
V      0 .No
V      1 .No (derived)
V      2 .Yes

```

1.5 Troubleshooting

Four environment variables must be appropriately set before running the Population Synthesizer:

- `TRANSIMS_HOME` - root directory of the TRANSIMS distribution. The directory `TRANSIMS_HOME/data/synpop` must be present and have read permission by the user.
- `STF_INFO_DIR` - Set to `TRANSIMS_HOME/data/synpop/Parep2/stf`. This directory must be present and have read permission by the user.
- `STF_DATA_DIR` - Set to directory where STF3A data resides. Sample STF3A data is in `TRANSIMS_HOME/data/synpop/stf` and `STF_DATA_DIR` should be set to this directory if using New Mexico PUMA 00400. If using data from a mounted cdrom, set `STF_DATA_DIR` to the CDROM directory (Example: `/mnt/cdrom`). The directory must be available and have read permission by the user.
- `PUMS_DATA_DIR` - Set to directory where PUMS data resides. Sample PUMS data for New Mexico is in `TRANSIMS_HOME/data/synpop/pums` and `PUMS_DATA_DIR` should be set to this directory if using New Mexico PUMA 00400. If using data from a mounted cdrom, set `PUMS_DATA_DIR` to the CDROM directory (Example: `/mnt/cdrom`). The directory must be available and have read permission by the user.

The directory `TRANSIMS_HOME/data/synpop/doc` must also be available and have read permission by the user.

Each PUMA should be processed separately. After entering a PUMA number in the PUMA ID window, you must press Enter (Return) for the entry to be processed. The PUMA ID that is entered must exactly match the PUMA number in the MABLE/GEOCORR file and the PUMS data, i.e. 00400 not 400.

1.6 Simplified Population Generator

Populations and activities for the small Multimode Network (see Section 3.3 in this volume) are produced using simplified generators. These generators imitate all of the characteristics of the true generators. Single person households are created and placed on the *walk* layer of the Multimode Network. These are located according to the land use associated with the network activity locations. Each person is assigned a vehicle and a vehicle starting position. The starting position of each vehicle is at the parking location associated with the activity location of the home.

The activity location file of the Multimode Network has three columns of land use data. These are denoted by A (Access to Transit), H (Home), and W (Work). The column A is the walking distance in meters to the nearest transit stop, H is the weight given to siting a household at the activity location, and W denotes the attractiveness for work activities at a particular activity location. In the following, we let the symbols, A_i , H_i , and W_i , be the value of these variables at the i^{th} activity location.

The simplified population has the required characteristics of the populations as outlined in Section 2.2.2.1 of Volume 3—*Files*. The simplified population generator makes single family households where the household records contain a census tract and block group number, which are both set to

0, a unique household ID, one vehicle, one person and a household location. The person files contain the household ID, a unique person ID, and a generic demographic that we denote as income.

The location on the *walk* layer of the network for each household is determined using the H_i values from the activity location file. Each activity location is assigned a probability, p_i of containing households. This probability is

$$p_i = H_i / \sum_k H_k$$

The household location is determined by drawing a random number between zero and one, say z , and assigning the household to activity location i if

$$\sum_{k=1}^{i-1} p_k < z \leq \sum_{k=1}^i p_k$$

TRANSIMS requires that each vehicle have a starting location. These locations are the parking lots on the *drive* layer of the network. In the Multimode Network each activity location is attached to one parking lot by a network process link. The starting location for the household vehicle is the parking location that is attached to the activity location of the household by a process link.

Individual incomes for the population are set to be between \$10,000 and \$100,000 by random draws from a scaled beta distribution. The beta probability density function is given by the function

$$\beta(a, b) = \frac{G(a, b)}{G(a)G(b)} x^{a-1} (1-x)^{b-1}$$

Income is generated by random draws from

$$90000 \beta(2, 3) + 10000$$

1.6.1 Usage

Two programs are used to produce a simplified synthetic population that is located on a transportation network. A TRANSIMS configuration file is used to specify parameters to the programs. Entries in the configuration file are of the form

<KEY> <value>

1.6.1.1 Generating a Simplified Baseline Population

Popgen is the program that generates a baseline population of single-person households with one vehicle per household. This population is not located on the transportation network and uses a value of -1 for the household location. *Popgen* assigns household and person IDs and generates income as the only person demographic in the population.

Usage: Popgen <configuration file>

Popgen uses the following configuration file keys from the TRANSIMS configuration file. Some keys have default values that may be used if the key is not specified in the configuration file.

Table 11: *Popgen* configuration keys.

Configuration Key	Description
POP_NUMBER_HH	The number of households to be generated.
POP_BASELINE_FILE	The name of the output file.
POP_STARTING_HH_ID	The number from which the generated households will be sequentially numbered. Default = 1.
POP_STARTING_PERSON_ID	The number from which the generated persons will be sequentially numbered. Default = 101.

1.6.1.2 Locating a Simplified Population on the Transportation Network

Two programs are available to locate a population on a transportation network: *Poploc* and *BlockGroupLoc* (See Sect 1.3.2). *Poploc* should be used when the baseline population was generated independent of census tract and block group information. The TRANSIMS Simplified Population Generator (*Popgen*) produces a population of this type.

Poploc is the program that locates a population on a transportation network based upon choice of a home location from residential land use values specified in the network activity location file (user data). All activity locations in the network that have residential values greater than zero are candidates for the selection. The baseline population should contain valid household and person IDs since *Poploc* does not generate them.

Usage: Poploc <configuration file>

Poploc uses the following configuration file keys from the TRANSIMS configuration file. Some keys have default values that may be used if the key is not specified in the configuration file.

Table 12: *Poploc* configuration keys.

Configuration Key	Description
POP_BASELINE_FILE	The name of the file containing the baseline population.
POP_LOCATED_FILE	The name of the file where the located population will be written.
ACT_HOME_HEADER	The user data column header in the network activity location file used to specify household home locations. Default = <i>HOME</i> .
NET_DIRECTORY	The directory where the network files reside.
NET_NODE_TABLE	The network node table name.
NET_LINK_TABLE	The network link table name.
NET_ACTIVITY_LOCATION_TABLE	The network activity location table name.

2. ACTIVITY GENERATOR MODULE

2.1 Overview

The TRANSIMS Activity Generator module generates a list of activities for each member of the synthetic population. Each activity consists of the type of activity, its priority, starting and ending time preferences, a preferred mode of transportation, a vehicle preference if appropriate, a list of possible locations for the activity, and a list of other participants if the activity is shared. The set of activities for each household is based on the household demographics. They form the basis for determining individuals' trip plans for the region, resulting in travel demand sensitive to the demographics of the synthetic population. The activities also are sensitive to the network. Activity locations reflect land use and employment data from the network.

The design of the Activity Generator is based on two principles. The first is to capture household behavior accurately, not just activity/travel patterns for individuals. Thus, if one family member takes a child to school, another need not do so. Household interaction is generated. The second goal is relative simplicity in the models for activity location. The location choice models are fairly simple. Instead of attempting to implement detailed models with network skim times, etc., feedback from the Traffic Microsimulator is used to refine activity location choices.

The activity generation algorithm has two steps. The first step uses a survey from the city to generate a library of skeletal activity travel patterns. These patterns consist of the actual records of activities, times, and modes—but stripped of locations. The survey households are divided into groups arranged in a binary tree using a regression technique keyed on household demographic variables as described in Speckman, Sun and Vaughn [1999]¹. The terminal nodes of the tree are chosen to be as homogenous as possible with respect to household activities. The demographic variables of a synthetic household place it in a unique node in the tree, and a matching survey household is chosen from that node at random. The skeletal household pattern from the survey household then becomes the basis for the activities in the synthetic household. Locations for the activities are chosen using multinomial logistic choice models. Work locations are chosen first, and other activities are added using a trip-chaining multinomial logistic model. The skeletal pattern provides all necessary information for household interaction including shared rides. This information is retained in the activity list output to the Route Planner.

2.2 Algorithm

The activity generation algorithm has four steps.

1. *Create skeletal patterns from the survey.* The activity lists for each survey household are organized by trips and stripped of locations to create a library of skeletal activity/travel patterns.
2. *Match synthesized households with survey households.* Each synthesized household is matched with a household from the survey. The matching is done using a tree-structured classification based on the demographics of the households. Each survey household is

¹ Speckman, P., Sun, D., and Vaughn, K.M., (1999) *Identifying Relevant Socio-Demographics for Distinguishing Household Activity-Travel Patterns: A Multivariate CART Approach.*

assigned to a unique node in the tree. After the synthesized household is assigned to its node, a survey household is chosen at random from the same node to obtain a matching household. The skeletal patterns for survey household members are then assigned to the matching members in the synthesized households.

3. *Generate locations by discrete choice models.* Because households are matched on demographics, an activity list for a person in the matching survey household is appropriate for a person in the reconstructed household except for activity location. New locations are generated for the activities from choice models using the synthesized household location.
4. *Group household members to trips.* The final activity set includes a list of participants for each activity.

The flow of the algorithm is given in Figure 7.

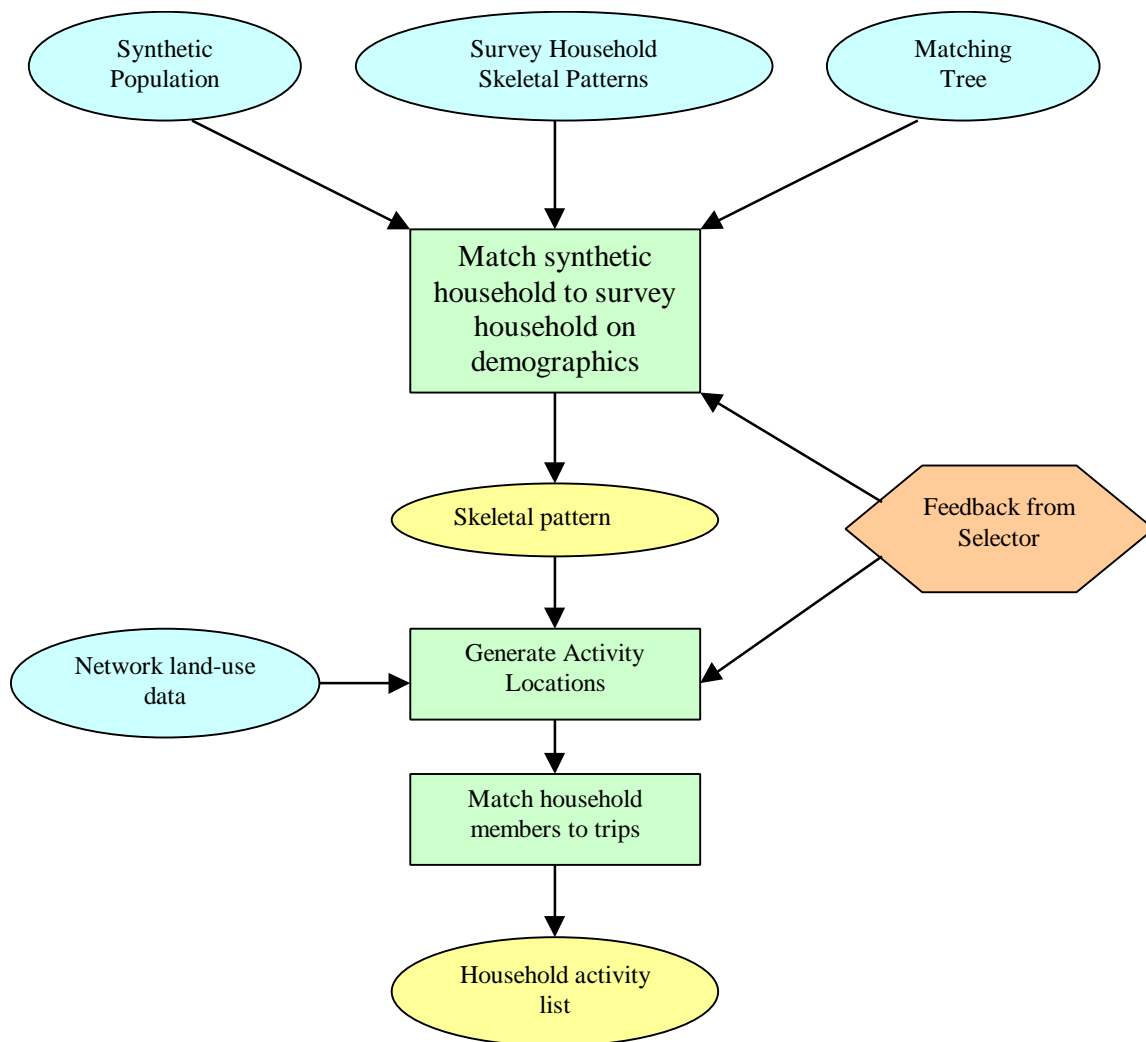


Figure 7: Activity Generator.

The synthetic population household demographics, survey household demographics, survey household activity/travel patterns, and a binary matching tree are inputs to the first stage, producing a skeletal activity/travel pattern for the synthesized household. Network land-use data

are then used to generate activity locations for the skeletal pattern, and household members are matched to trips to create the final activity list.

2.2.1 Binary Tree Matching

Cross-tabulation of households by demographic variables can easily create many cells with few or no households in the survey. Instead of matching through some kind of table, household matching is implemented by locating households in the terminal nodes of a binary tree. This tree is chosen to be sensitive to characteristics of household behavior but to be parsimonious with respect to household characteristics that do not affect behavior.

To illustrate a binary matching tree, consider the simple example given in Figure 8. (Actual trees used in generating activities are much more complicated.) This tree is constructed with an abbreviated list of three household demographic variables:

<i>hhsiz</i>	household size
<i>agelt5</i>	number in household aged less than 5
<i>age5to17</i>	number in household from age 5 to age 17

The tree has 13 nodes, including seven terminal nodes indicated by squares. At each non-terminal node, a household is classified further into either a left or right *child* node according to a simple rule given by a demographic variable and *split point*. If the appropriate variable is less than the split point, the household is classified into the left node; otherwise the right node is chosen. In the example below, the first choice (node 1) is on household size (*hhsiz*). If $hhsiz < 2.5$, the household falls somewhere in the left nodes. Otherwise, further classification proceeds to the right. The procedure continues recursively until a terminal node is reached.

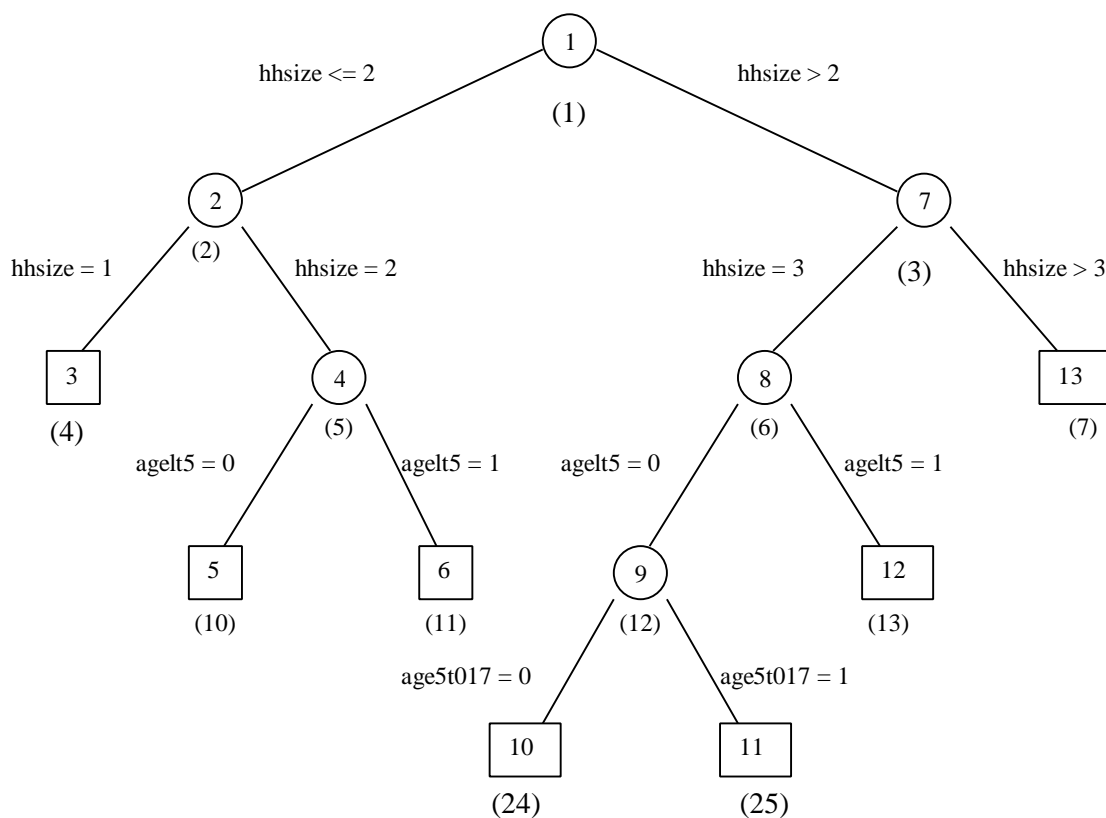


Figure 8: Sample household matching tree.

The tree has 13 nodes, six nonterminal nodes indicated by circles and seven terminal nodes indicated by squares. The numbers inside the squares and circles are node numbers. The numbers beneath the nodes in parentheses are binary node numbers defined in Section 1.6.2. The seven terminal nodes are defined in Table 13.

Table 13: Terminal nodes.

Node	Description
3	Household size = 1
5	Household size = 2, no children less than 5
6	Household size = 2, 1 child less than 5
10	Household size = 3, no children less than 5, no children between 5 and 17
11	Household size = 3, no children less than 5, 1 child between 5 and 17
12	Household size = 3, 1 child less than 5
13	Household size greater than 3

The first step in generating a set of activities is to locate the synthetic household in its unique terminal node of the tree. A survey household is then selected at random from the node. For flexibility, weights are used in choosing the survey household. Each survey household has weight w_i assigned to it in the Survey Weights file. If N is the terminal node for the synthetic household, survey household i in node N is chosen with probability

$$p(i) = \frac{w_i}{\sum_{j \in N} w_j}.$$

2.2.2 Matching Individuals Within the Household

Synthetic household members are matched to members in the survey household based on the following four demographic variables.

<i>relate</i>	Variable coded for relation to head of household 1 = spouse or partner, head of household PUMS variable RELATE1 = 00, 01, 10 2 = child RELATE1 = 02, 03, 06 3 = adult relative RELATE1 = 04, 05, 07 4 = other
<i>work</i>	work status 1 = full-time/part-time worker (including self-employed) 2 = nonworker
<i>gender</i>	1 = male 2 = female
<i>age</i>	age in years

These four variables MUST be the first for person demographics in both the synthetic population file and the survey demographic file.

Perfect matches are not always possible; the Activity Generator attempts to find the best matches possible and sets a flag to resample from the pool of survey households in the matching node if certain conditions hold. First, children are matched to children and adults are matched to adults. The children in the synthetic household are sorted by gender and age (descending), the children in the survey household are sorted in the same way by gender and age, and a one-to-one match is made between the two sorted lists. If the survey household has more children than the synthetic household, the extra children in the survey household are ignored. If the synthetic household calls for more children than the survey household has, the activities of the last child in the survey household are replicated as often as necessary to create activities for the remaining children in the synthetic household. If there are no children in the survey household but the synthetic household has children, the error flag is set to resample from the survey households and try a different match.

The rules for matching adults are similar. The adults in the synthetic household and the survey household are sorted by the variables relate, work, age (descending sort) and gender. Again, a one-to-one match is made. If the synthetic household contains more adults than the survey household, the activities of the last adult survey household member are replicated as often as necessary. At present, no further error checking is performed in the matching algorithm for adults.

2.2.3 Multivariate Regression Tree Program

The binary tree for matching synthetic households to survey households is constructed with the aid of a multivariate regression tree program. A brief description is given here. Details are contained in Vaughn, Speckman and Sun [1999]².

A regression tree is a technique for modeling a regression relationship between a dependent variable Y and independent variables X_1, X_2, \dots, X_p . Regression trees are useful when there are a large number of explanatory variables and a complex relationship between the response variable and the explanatory variables is expected. In these cases, tree-based methods may be better able to capture non-additive behavior and be easier to interpret than linear models. The CART algorithm (Classification and Regression Trees) was introduced by Breiman et al. [1984]³ and has been implemented as the tree function in the S-PLUS software package (see Clark and Pregibon [1990]⁴). The basic idea is to partition the data set into nodes defined by the predictor variables X_1, X_2, \dots, X_p , and to model the response as a constant within each node. The nodes are defined by a binary tree.

Tree modeling, as implemented in the CART algorithm, typically consists of two stages: a forward recursive algorithm for “growing” the tree, and a second stage where the tree is “pruned back.” Because the “growing” process is only in the forward direction (once a node is defined, it cannot change) the algorithm does not necessarily produce an optimal tree. The strategy is to begin by growing a very large tree—one that probably overfits the data—and then to use a second algorithm balancing faithfulness to the data with the complexity of the tree to select a good subtree. The philosophy is related to forward selection in the usual regression setup, where a liberal rule is adopted in the entering of variables to ensure that no important variables are omitted.

2.2.3.1 CART Algorithm

2.2.3.1.1 Growing the Tree

To define the recursive algorithm, consider observations in a single “parent” node P_i as part of tree T . At the next stage, the parent node will be split into two “children” nodes, a left node, L_i , defined as all i' observations in P_i with $X_{ij} \leq t$ and a right node, R_i , where $X_{ij} > t$ for a suitable choice of variable X_{ij} and cut point t . The optimal variable and cut point for the split are defined in terms of the “deviance” of the node, given as

$$D(N) = \sum_{i \in N} (Y_i - \bar{Y})^2,$$

the corrected total sum of squares for the observations in the node. For a potential partition split on variable X_j at cut point t , define the reduction in deviance from the split as

² Vaughn, K.M., Speckman, P. and Sun, D. (1999) *Identifying Relevant Socio-Demographics for Distinguishing Household Activity-Travel Patterns: A Multivariate CART Approach*.

³ Breiman, L., J.H. Friedman, R.A. Olshen and C.J. Stone. (1984) *Classification and Regression Trees*. Chapman and Hall, London.

⁴ Clark, L.A., and Pregibon, S. (1990) *Tree-based models*. In Statistical models in S. Wadsworth and Brooks/Cole, pp 377-419.

$$\Delta_{j,t} = D(P) - (D(L) + D(R)).$$

A search is conducted over all j and t to find the pair j^*, t^* such that

$$\Delta_{j^*, t^*} = \max_{j,t} (\Delta_{j,t}),$$

subject to,

1. $i', i'' > \text{some minimum (say 10)}$
2. $D(P) > 0.01 * D(\text{total})$,

where $D(\text{total})$ is the deviance in the dependent variable before any splits are made. If either condition fails, the parent node is a terminal node. The algorithm continues recursively splitting nodes until all nodes are terminal nodes.

2.2.3.1.2 Pruning the Tree

Prediction for a regression tree is simple. The dependent variable Y is estimated by the mean value of Y in each node. However, the binary tree from the growing algorithm generally overfits the data. Several proposals have been made to determine a better tree. One common way to assess how well a tree fits is by using it to predict a new set of data. In this case, deviance is replaced by sum squared prediction error, and the best subtree in the sense of minimizing prediction error can be determined. However, holding a subset aside for validation may be wasteful, and the tree selected depends partially on the set of data selected to be held out. Following Breiman et al. (1984), S-PLUS implements a form of cross-validation designed to emulate this kind of validation process without wasting data.

The data set is randomly partitioned into ten approximately equal parts. Each part is held out in turn. A subtree T' is then re-estimated on the remaining 90% of the data, and the re-estimated tree is used to forecast the 10% held out. Let, $CV_i(T')$ denote the sum squared prediction error for the i th partition. The process is repeated for all ten subsets of the data, and a total cross-validation score

$$CV(T') = \sum_i CV_i(T'),$$

is computed for the subtree. A subtree that minimizes (or nearly minimizes) $CV(T')$ is a good final choice for a tree that is appropriate for the data.

2.2.3.2 Multivariate Tree Algorithm

Our implementation of the multivariate regression tree uses the following extended definition of deviance. Suppose we have dependent variables Y_1, \dots, Y_d and node N with i' observations. Let the deviance at node N with respect variable Y_j be given as

$$D_j(N) = \sum_{i'} (Y_{ij} - \bar{Y}_j)^2,$$

with the total deviance at node N

$$D(N) = \sum_j s_j D_j(N) = \sum_j \sum_{i'} s_j (Y_{ij} - \bar{Y}_j)^2,$$

where $s_j = 1/\text{var}(Y_j)$ and is a scale factor.

Then for a tree T ,

$$D(T) = \sum_j s_j D_j(T) = \sum_j \sum_i s_j (Y_{ij} - \bar{Y}_j)^2$$

is the scaled total sum of squares for the i observations in the tree. Nodes can now be split using total deviance instead of the single variable deviance. With this new definition of total deviance, the regression tree can be grown and pruned as before. Cross-validation can be used with this definition of deviance to prune the tree to a proper size.

2.2.4 Activity Location Generation

The following steps summarize the location generation algorithm. Locations are chosen in two stages. A discrete choice model is first used to select appropriate zones for all activities. Land-use variables are then used to find specific activity locations within zones for each activity.

- 1) Generate all work locations with a discrete choice model.
- 2) Divide activities into trips:
 - Home to home
 - Home to work
 - Work to home
 - Work to work
- 3) Generate locations for other activities for drivers using trip-chaining discrete choice models.
- 4) Match other household members to trips.

2.2.4.1 Work Location Model

The work location model is a simplified multinomial logistic choice model, defined with the following terms.

L	Destination zone for work activity.
$a(L)$	Attractor for work activity in zone L . Often $a(L) = c'x(L)$, where $x(L)$ is a vector of land use variables for zone L , and c is a coefficient vector fit by maximum likelihood. It is also possible to use other specifications for $a(L)$ including a nonparametric model for a continuous distribution fit from data. This model is described in a technical report by Speckman, Sun and Vaughn [1998].
$d(H,L)$	Distance from home location H to work location L .
b_m	Coefficient for mode choice m

The destination zone is chosen according to the probability distribution

$$p(L) = \frac{\exp(a(L) + b_m d(H, L))}{\sum_{L'} \exp(a(L') + b_m d(H, L'))}.$$

Mode choice is taken from the survey household skeletal pattern. After the zone is chosen, a specific activity location within the zone is chosen at random according to weights given in the Activity Location file. For work, total employment w_A for each location is used, and location A is chosen with probability

$$p(A) = \frac{w_A}{\sum w_{A'}},$$

where the sum is taken over all locations A' in zone L .

The model is simple for several reasons. Since TRANSIMS starts with an empty network, the activity generator does not have access to realistic travel times. At present, there is no mechanism to feed back network skim times from the Traffic Microsimulator to the Activity Generator to refine the location choice probability model. Instead, distance is used as a surrogate for time. If the Route Planner or the Traffic Microsimulator detects an impossible set of activities, new locations can be generated or the entire household set of activities can be replanned. The feedback mechanism to replan activities is under development.

2.2.4.2 Locations for Other Activities

A logistic multinomial choice chaining model is used to generate locations for other activities. To illustrate, consider a trip chain consisting of two stops on the way from work to home. Suppose the skeletal pattern calls for travel from work location L to “visit” at L_1 by mode m_1 , a second stop to “shop” at location L_2 by mode m_2 , finally returning home by mode m_3 . The home and work locations are known. The other two destination zones, L_1 and L_2 , are drawn from the joint probability distribution

$$p(L_1, L_2) = \frac{\exp(b_{m1} d(L, L_1) + a(L_1, v) + b_{m2} d(L_1, L_2) + a(L_2, s) + b_{m3} d(L_2, H))}{\sum \sum \exp(b_{m1} d(L, L'_1) + a(L'_1, v) + b_{m2} d(L'_1, L'_2) + a(L'_2, s) + b_{m3} d(L'_2, H))},$$

where the double sum is over all pairs of zones (L_1, L_2) . Separate attractors $a(L, t)$ are defined for each location L and activity type t , where t is either v for “visit” or s for “shop” in this example.

2.2.4.3 Example

As an example, a set of activities was generated for a household in Portland, Oregon with two adults. The node in the matching tree consisted of all households with two adult workers. There were 656 survey households in the node, and one was selected at random as a match. The first person had a single work activity and then made a stop to visit, a stop to shop, a maintenance stop (car wash), and returned home. The set of generated activities is shown on a map with the EMME/2 Portland network in Figure 9.

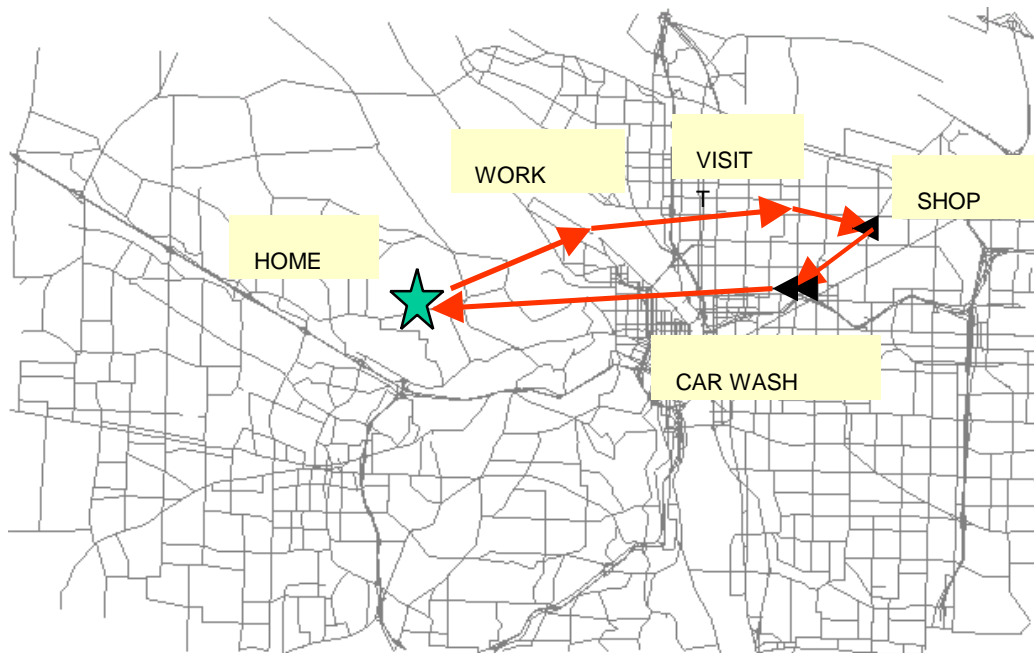


Figure 9: Activites for first traveler.

The second person went from home to work with a stop on the way, then went out of town on a business trip (coded “other work” in the survey), returned to his office, and went home. The activity list currently has no code for out-of-town business, so as default, the Activity Generator chose a second work location for the “other work” activity. The survey also had incomplete information on the last work activity with a missing location. The activity generator created a third location for this activity. The resulting activity pattern is shown in Figure 10.

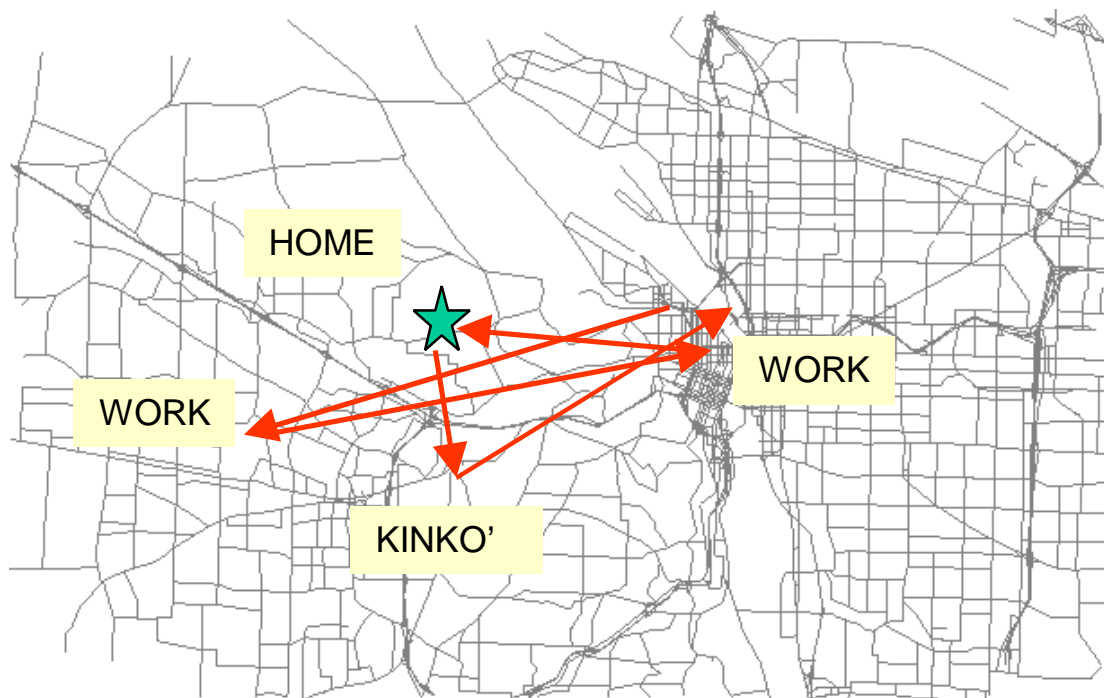


Figure 10: Activities for second traveler.

This example shows that the Activity Generator can function even with missing data in the survey record. The correct number of trips was generated.

2.2.5 Activity Times and Durations

In keeping with the spirit of resampling, activity times are taken from the skeletal activity pattern with as little change as possible for out-of-home activities. The format for specifying activity times is described in Volume 3—*Files*. Each activity has a preferred start time, end time, and duration. The range of each of these times is specified by a beta distribution requiring four parameters—lower bound L , upper bound U , and “shape” parameters a and b . When $a = 1$ and $b = 1$, no preference is indicated within the range L to U . If $a = -1$ and $b = -1$, the range is assumed to be 0 around the preferred time. Preferred times are now taken directly from the skeletal patterns. Table 14 gives the values of the remaining parameters as currently implemented in the Activity Generator.

Table 14: Settings of time parameters for activities. *Obs.* means observed time taken from skeletal pattern. The times are in hours.

Type of activity		L	U	a	b
Work	Start	Obs. - .25	Obs. + .25	1	1
	End	Obs. - .25	Obs. + .25	1	1
	Duration	Obs. - .25	Obs. + .25	1	1
Other out-of-home	Start	Obs. - .50	Obs. + .50	1	1
	End	Obs. + .50	Obs. + .50	1	1
	Duration	Obs. - .3(obs.)	Obs. + .3(obs.)	1	1
AM beginning at-home	Start	0	0	-1	-1
	End	Obs. - .75	Obs. + .75	1	1
	Duration	Obs. - .75	Obs. + .75	1	1
Home during day	Start	Obs. - .75	Obs. + .75	1	1
	End	Obs. - .75	Obs. + .75	1	1
	Duration	Obs. - 1.00	Obs. + 1.00	1	1
PM end at-home	Start	Obs. - .75	Obs. + .75	1	1
	End	24.00	24.00	-1	-1
	Duration	Obs. - .75	Obs. + .75	1	1

2.3 Usage

The Activity Generator uses data files that are specified in a configuration file. The data files are specific to a network and population and use patterns derived from the Portland activity survey. Data files are supplied with the TRANSIMS distribution for the Local Streets Network only.

The data files are in the *TRANSIMS_HOME/data/acts* directory. The file names are specified in the Activity Generator configuration file, *TRANSIMS_HOME/config/atp.config*. The configuration file is automatically edited during the TRANSIMS installation to point to the data files in the *TRANSIMS_HOME/data/acts* directory. Refer to Section 2.6.1 for a description of the configuration file contents.

The Activity Generator program is *TRANSIMS_HOME/bin/ActivityGenerator*. The name of the configuration file is given as a command line argument when the program is invoked.

```
% $TRANSIMS_HOME/bin/ActivityGenerator $TRANSIMS_HOME/config/atp.config
```

The Activity Generator produces a file containing the activity set for the population as well as a log file containing diagnostic information about the run.

2.4 Tutorial

The environment variable `TRANSIMS_HOME` must be set to the directory where the TRANSIMS distribution is installed.

- 1) Change directory to *TRANSIMS_HOME/output*
 % `cd $TRANSIMS_HOME/output`
- 2) Run the Activity Generator using data files derived from a test population located on the Local Streets Network. The test population is in *TRANSIMS_HOME/data/synpop/local_pop.located*. The Activity Generator uses a TRANSIMS vehicle file to specify vehicle IDs that are used in the activities. The vehicle file is *TRANSIMS_HOME/data/vehicles/local.vehicles*. The Activity Generator will use the configuration file *TRANSIMS_HOME/config/atp.config*
 % `$TRANSIMS_HOME/bin/ActivityGenerator $TRANSIMS_HOME/config/atp.config`

The activity file, *local.act*, is produced by the Activity Generator in your current directory (*TRANSIMS_HOME/output*). Refer to Volume 3—*Files*, Section 3.3.5, for a description of the format of a TRANSIMS activity file. The initial mode choice given in the activity file is always car mode for the Local Streets Network. The mapping between the integer mode value in the activities (1) and the mode string used by the Route Planner is given in a modes file, *TRANSIMS_HOME/data/modes/local.modes*.

The Activity Generator also produces a diagnostic log file in your current directory. The name of the log file (*atp.log*) is specified in the configuration file.

2.5 Troubleshooting

The Activity Generator has data files that will work with only the Local Streets Network and the test population located on that network. It cannot be used on other networks in the distribution.

The Activity Generator uses the system time to seed the random number generator which causes a slightly different set of activities to be generated for each run.

2.6 Files

The Activity Generator uses a number of files.

2.6.1 Configuration Files

Input to the Activity Generator is controlled through the file *atp.config*. Each line has the form

Keyword file_name

Both the keyword and the file_name must appear on the same line. The definitions shown in Table 15 are required.

Table 15: Configuration keywords.

Configuration Key	Description
ATP_log	log file containing diagnostic information for run
ATP_out	output file of activities
ATP_demo_tree	binary tree for matching survey households to synthetic households
ATP_activities	activity lists for the survey households.
ATP_survey_weights	weights used in sampling to match survey households to synthetic households
ATP_survey_demo	survey household demographic file
ATP_loc_taz	zone definitions including attractors for activities
ATP_loc_emme2	network file
ATP_loc_weights_mode	coefficient matrix for choice models
ATP_synth	synthetic population file
ATP_vehicles	vehicle file
ATP_synth_gen	list of synthetic household ids. Activities are generated for each household in the list.

Example:

ATP_log	log
ATP_out	atp.out
ATP_demo_tree	data/tree2.dat
ATP_activities	data/survey.activities
ATP_survey_weights	data/surv_weights.dat
ATP_survey_demo	data/samp_demog.dat
ATP_loc_taz	data/zone.dat
ATP_loc_emme2	data/activity.locations
ATP_loc_weights_mode	data/mode_weight.dat
ATP_synth	data/pop_hh_per.dat
ATP_vehicles	data/pop.vehicles
ATP_synth_gen	data/portpop.samp

2.6.2 Binary tree

The tree structure is specified by three files. The demographic variables used to define the tree are assumed to be the variables <HHdata1 ... HHdatan> in the synthetic population file. The order of appearance in the synthetic population file determines the numbering of the variables in the tree, and this order must match the variables in the survey population file. The third specification file gives the tree structure itself. In the example, the variables are

```
Household demographics:    hhsize alt5 a5to17
```

These variables may come directly from census classifications or may be constructed from the survey and synthetic population household member data records.

The tree definition is given by the following file.

Terms

Node number	Sequential node number. Determined recursively starting with 1 for the root node and proceeding to the left until a terminal node is reached.
Variable	Demographic variable for splitting node into two children nodes. Coded 0 for terminal node.
Split value	Value used along with the split variable to create children nodes. Coded 0 for terminal node.
Binary node number	Node number derived recursively as follows. If binary node number k is split into two nodes, the left one is number $2k$ and the right is number $2k + 1$.
Right node number	Sequential node number for right node of split. Coded 0 for terminal node.

The file format has one line for each node with the format:

```
node_number variable split_value binary_node_number right_node_number
```

Example:

The simple tree in Section 2.2.1 assumes the following demographics for both the synthetic population and the survey population files.

```
Household demographics:    hhsize alt5 a5to17
```

The following file gives the complete specification of the tree.

```
1 1 2.5 1 7
2 1 1.5 2 4
3 0 0 4 0
4 1 0.5 5 6
5 0 0 10 0
6 0 0 11 0
7 1 3.5 3 13
8 2 0.5 6 12
9 3 0.5 12 11
10 0 0 24 0
11 0 0 25 0
12 0 0 13 0
13 0 0 7 0
```

2.6.3 Survey Weights

Sampling to obtain a matching survey household is described in Section 2.2.1. The file containing the weights has one entry per line, a decimal number. The length of the file must match the number of survey households, and each line corresponds to a survey household in sequential order.

2.6.4 Survey Demographic Data

The activity generator uses a set of demographic data for the survey population corresponding closely to the data for the synthetic population.

2.6.4.1 File Format

The survey file contains two header lines, followed by data lines.

2.6.4.1.1 Header Lines

The first line of the file contains the household demographic and user data information. The second line of the file contains person demographic information.

2.6.4.1.2 Format

The format of the lines is:

```
<text>: <description demog/data1> . . . <description demog/dataN>
```

The <text> entry may be any text comment that is meaningful to the user. The <text> entry MUST be followed by a colon (:). A single word description of each of the household demographic/data or person demographics in the file follows the colon. Each household data item that is present in the household data lines of the file must have a single word description in the household header line (line 1 of the file). Each person demographic that is present in the person data lines of the file must have a single word description in the person demographic line (line 2 of the file). The single word description must not contain white space.

The household demographics demographic/data variables are the variables used for the tree matching algorithm and MUST agree with the demographic/data variables in the synthetic population file. The person demographics must have the variables relate, work, gender and age as defined in Section 1.2.2.

2.6.4.1.3 Example

```
Household Demographics: HHSIZE ALT5 A5TO17  
Person Demographics: RELATE WORK GENDER AGE
```

The household data lines in a file with this header will have three household demographic values (HHSIZE, ALT5 and A5TO17). The person demographic data lines will have the four required variables RELATE, WORK, GENDER and AGE.

2.6.4.2 Data Lines

The data for a survey household span multiple lines in the survey demographics file.

2.6.4.2.1 Format

The first line of a household record contains the household data:

```
H <SURVEY HH ID> NUMVEH <HHData1> ... <HHDataN>
```

The SURVEY HH ID is a unique identifier for the survey household, and NUMVEH is the number of vehicles in the household.

Following the household data are N lines, where N = number of persons in the household, of person data.

```
<SURVEY HH ID> P <SURVEY Person ID> RELATE WORK GENDER AGE
```

2.6.4.2.2 Example

Household 200120 has 2 vehicles, 4 persons, no one age less than 5, and two children aged 5 to 17.

```
H 200120 2 4 0 2
200120 P 3 1 1 1 40
200120 P 4 1 1 2 42
200120 P 5 2 2 2 9
200120 P 6 2 2 2 6
```

2.6.5 Zone Data

The zone data contains geographic data for the zones used in the Activity Generator and the attractors by zone and activity type used in the location choice models.

2.6.5.1 Format

There is one header line with definitions of the variables. Each variable has a one-word descriptor with no white space. The format is as follows.

```
zone NORTHING EASTING WATTRACT <attractor1> ... <attractorN>
```

The attractor variables MUST correspond to the activity definitions in the generator. The first attractor WATTRACT is for work activities. The remaining attractors are for activities 1 – N

2.6.5.2 Example

The following file gives zone data for a test network with 10 zones. Six activity types are defined.

ZONE	EASTING	NORTHING	WORK	SHOP	SCHOOL	VISIT	OTHER	SERVE_PASS
1	50	16050	2	2	-10	-10	2	1
2	50	19050	2	2	-10	-10	2	1
3	16050	35050	2	2	-10	-10	2	1
4	19050	35050	2	2	-10	-10	2	1
5	35050	19050	2	2	-10	-10	2	1
6	35050	16050	2	2	-10	-10	2	1
7	19050	50	2	2	-10	-10	2	1
8	16050	50	2	2	-10	-10	2	1
9	17500	17500	-10	-10	-10	-10	-10	1
10	17500	17500	-10	-10	3.71	3.71	-10	1

2.6.6 Mode Coefficient Data

The coefficients b_m for distance in the choice models of Section 2.2.4 depend on mode m . The file containing these coefficients currently defines modes for eight mode choices:

- 1) Other
- 2) Walk
- 3) Bicycle
- 4) School Bus
- 5) Public Bus
- 6) Light Rail
- 7) Personal Vehicle
- 8) Non-personal Vehicle

The control file currently has two columns—each with 8 entries. The first column contains coefficients for work activities. The second contains coefficients for all other activities.

Example:

```
-0.00008    -0.00008
-0.00008    -0.00008
-0.000109   -0.000109
-0.000085   -0.000085
-0.0000535  -0.0000535
-0.0000760  -0.0000760
-0.000040   -0.000040
-0.0000535  -0.0000535
-0.000046   -0.000046
```

2.6.7 Survey Activities

Skeletal household activity patterns are generated from a data set of activities from the survey households. The list of activities must match the survey household demographics file. Each household in the demographics file must appear in the same order with activities in the activity file.

2.6.7.1 File Format

Activities are grouped sequentially for each survey household. The file is an ASCII file with entries separated by one or more spaces. Missing values are possible for some of the variables. The missing value code is “.”.

Table 16: Activities file format.

Field	Description	Allowed Values
Survey Household ID	Each survey household has a unique ID	integer
Person Number	Each person in the household has a unique ID, starting with 1. Numbers are only unique within the household.	integer: 1 through household size
Activity Number	Activity number for each person.	integer: 0 through n: 0 = initial at-home activity of the day if necessary

Field	Description	Allowed Values
Activity Type	Definitions may vary. Currently the following definitions are required. 0 = at-home activity 1 = work 22 = serve passengers Others may vary.	integer: 0 through n: Example: 0 = at-home activity 1 = work 2 = shop 3 = school 4 = visit 5 = other 22 = serve passenger
At Home	Coded 1 for activity at-home, 2 for out of home	integer: 1 or 2
Were-you-there	Coded 1 if person was already at the location, 2 if not. Missing value code "." is allowed.	integer: 1 or 2 or "."
Mode for arriving at activity	Integer code for mode. Missing value "." allowed for if no travel to or from activity.	integer: 1 through n or ".": 1 = Other 2 = Walk 3 = Bicycle 4 = School Bus 5 = Public Bus 6 = Light Rail 7 = Personal Vehicle 8 = Nonpersonal Vehicle
Driver	Coded 1 if person was the driver, 2 if passenger, "." otherwise.	integer: 1, 2 or "."
Activity Start Time	Time of start of activity in minutes after midnight.	integer: 0 through 2400
Activity End Time	Time of end of activity in minutes after midnight.	integer: 0 through 2400
Number in Vehicle	Number of people in vehicle	integer: 1 through n or "."
Geocode x	Easting geocoordinate. Must be in units (e.g. state-plane-feet) agreeing with mode coefficients.	decimal or "."
Geocode y	Northing geocoordinate. Must be in units (e.g. state-plane-feet) agreeing with mode coefficients.	decimal or "."

2.6.7.2 Example

The following set of activities are for household # 200087, a household with two persons. The first person has nine activities including an initial at-home activity. Activities 1 through 6 are a tour of out-of-home activities. The second person has one out-of-home activity.

SAMPNO	PERSNO	ACTNO	ACTID	AT_HOME	WUTHERE	MODE	DRIVER	NUMVEH	ACTSTART	ACTEND	EASTING	NORTHING
200087	1	0	0	1	180	480	7637347.0000	687428.0625
200087	1	1	5	2	2	7	1	1	480	630	7648077.5000	713295.7500
200087	1	2	5	2	2	7	1	1	660	715	7640385.5000	683294.6875
200087	1	3	5	1	2	7	1	1	730	760	7637347.0000	687428.0625
200087	1	4	5	1	1	.	.	.	765	915	7637347.0000	687428.0625
200087	1	5	4	1	1	.	.	.	930	1170	7637347.0000	687428.0625
200087	1	6	5	2	2	7	1	2	1200	1320	7639759.0000	685434.6875
200087	1	7	0	1	2	7	1	2	1335	1440	7637347.0000	687428.0625
200087	1	8	0	1	1440	1620	7637347.0000	687428.0625
200087	2	0	0	1	180	360	7637347.0000	687428.0625
200087	2	1	5	2	1	.	.	.	360	108	.	.
200087	2	2	0	1	1080	1620	7637347.0000	687428.0625

2.7 Simplified Activity Generator

Activities for the small Multimode Network (see Section 3.3 of this volume) that has activity locations around a circle are produced using a simplified generator that has all of the characteristics of the true activity generator. This Activity Generator makes a full activity file as given in Section 3 of Volume 3—*Files*. Activities are placed at the activity locations on the network, where all the activity locations are located on the *walk* layer of the small Multimode Network. Each individual in the simplified population for this network is assigned three activities. The individual starts with an activity at home, followed by one at work, and then a return activity at home. Three options are given for the location of the work activity. A mode choice model is also given with multiple options for determining the mode of travel.

2.7.1 Determining the Work Location

There three options to determine the work location.

Option A is based on a general gravity method and has input parameters, **a**, **b** and **g** all greater than 0. Let d_{ij} be the distance from activity location i to activity location j . Then for a fixed home location, i , the following weight h_{ij} is computed as:

$$h_{ij} = I(W_j) \{ \beta (\text{Exp}(-\alpha d_{ij} / W_j) + \gamma) \}$$

Where W_j is the work entry from the activity location file and both $I(W_j)$ and h_{ij} equal 0 if W_j is 0. Then the probability of a location j being the work location is

$$p_j = h_{ij} / \sum_k h_{ik}$$

The work location is determined by drawing a random number between zero and one, z , and choosing activity location i as the work location if

$$\sum_{k=1}^{i-1} p_i < z \leq \sum_{k=1}^i p_i$$

Option B fixes the work location at the activity location that allows work and is the maximum distance from the home location. Since the activity locations are in a circle, the home and work locations are as far apart on the circle as is possible. In mathematical terms the work location is at the location j where W_j is not equal to zero and d_{ij} , the distance from the home location i and location j , is a maximum when traveling around the circle in one direction.

Option C for obtaining the work location is identical to Option A except that the home to work distance, d_{ij} , is computed in one direction around the circle only.

☞ Only Option A is implemented for the June release.

2.7.2 Determining the Activity Times

All times in the activity file are in hours and fractions of hours. We assume that the first home activity starts at time 0 and that all work activities start between 0.5 and 3.5 hours (1,800 and 12,600 seconds) after time 0. Also the work duration is between 2.778 and 5 hours (10,000 and

18,000 seconds). The time spans in the activity file are all plus or minus 5 minutes or 0.083 hours. The parameters a and b in the file are set to -1.

The time to start work, T_w , depends on two input parameters, a and b . Other activity times, L_H , the time to leave home, D , the work duration, T_L , the time to leave work and T_H , the expected arrival at home are based on the initial time to start work, T_w . These quantities are computed as follows:

- 1) The time to start work: $T_w = (10800 \beta(\alpha, \beta) + 1800)/3600$, where $\beta(\alpha, \beta)$ represents a random draw from a beta density with parameters a and b as described in the section on simplified population generation.
- 2) The time to leave home: $L_H = T_w - d_{ij} / (4.8 * 7.5 * 3600)$. Note that $d_{ij} / (4.8 * 7.5 * 3600)$ is the average time in hours that it takes to travel on a non-congested road between parking locations i and j . “Cells” in the microsimulation are 7.5 meters long and the average speed is 4.8 “cells” per second.
- 3) The work duration: $D = (8000 * z + 10000)/3600$, where z is a random number between zero and one.
- 4) The time to leave work: $T_L = T_w + D$.
- 5) The time to arrive home: $T_H = T_L + d_{ij} / (4.8 * 7.5 * 3600)$.

These quantities are given in Table 17.

Table 17: Activity time table.

Activity	Start-a	Start-b	End-a	End-b	Duration-a	Duration-b
Home	0	0	$L_H - .083$	$L_H + .083$	$L_H - .083$	$L_H + .083$
Work	$T_w - .083$	$T_w + .083$	$T_w - .083 + D$	$T_w + .083 + D$	$D - .167$	$D + .167$
Home	$T_H - .083$	$T_H + .083$	24.	24.	$24 - T_H - .083$	$24 - T_H + .083$

2.7.3 Determining the Travel Modes

The following three mode sequences are allowed for the activities on the small Multimode Network:

- 1) **w**: Walk only.
- 2) **wcw**: Walk-Car-Walk.
- 3) **t**: Any combination of walk and bus.

The simplified Activity Generator has three options for determining the modes. One of these options, Option C, is a parameterized mode choice model where accessibility to transit is considered. The other options force driving or randomly picking the modes.

Option A: In this option, all modes are mode sequence 2, **wcw**.

Option B: One of the mode sequences **w**, **wcw**, or **t** is picked at random with this option.

Option C: In this option, a simplified mode choice model is used to pick either the mode sequence **wcw** or **t**. It is based on the traveler's accessibility to transit. The activity location file for the Multimode Network contains a column denoted A for accessibility. The entries in this column, A_i , are the walking distances in meters to the nearest transit stop.

The mode choice model depends on two user supplied parameters, $\alpha \geq 0$ and $0 \leq \beta \leq 2$. If the traveler is traveling between activity location i and activity location j , The probability of his taking transit (mode sequence 3) is given by the equation

$$p = \beta / (1 + \exp\{\alpha (A_i + A_j)\})$$

The choice between mode sequence 2, **wcw**, and mode sequence 3, **t**, is then determined by drawing a random number, z , between zero and one. If z is less than p , mode sequence 3 is chosen. Otherwise, mode sequence 2 is picked.

2.7.4 Usage

Two programs generate or modify simplified activities. The first, *Actgen*, generates simplified activity sets for all persons in the synthetic population. The second, *ActRegen*, modifies certain attributes of activities for the travelers specified in a TRANSIMS feedback file.

Both programs use the following configuration file keys from a TRANSIMS configuration file.

Table 18: *Actgen* and *ActRegen* common configuration keys.

Configuration Key	Description
POP_LOCATED_FILE	The name of the file containing the located population.
VEHICLE_FILE	The name of the TRANSIMS vehicle file for the population.
ACT_WORK_LOC_ALPHA	The alpha value used to generate work locations. Default = 1.0.
ACT_WORK_LOC_BETA	The beta value used to generate work locations. Default = 1.0.
ACT_WORK_LOC_GAMMA	The gamma value used to generate work locations. Default = 1.0.
ACT_TIME_ALPHA	The alpha value used to generate activity times. Default = 1.0.
ACT_TIME_BETA	The beta value used to generate activity times. Default = 1.0.
ACT_MODE_ALPHA	The alpha value used to generate activity mode choice (gravity method). Default = 1.0.
ACT_MODE_BETA	The beta value used to generate activity mode choice (gravity method). Default = 1.0.
ACT_WORK_LOCATION_OPTION	The option used to select the work location algorithm (1 - 3). Options 2 and 3 are not implemented in this release. Default = 1.
ACT_MODE_CHOICE_OPTION	The option used to select the mode choice algorithm (1 - 3). Default = 2 (random mode choice).
ACT_HOME_HEADER	The column header in the activity location file for single family home locations. Default = <i>HOME</i> .
ACT_WORK_HEADER	The column header in the activity location file for work locations. Default = <i>WORK</i> .
ACT_ACCESS_HEADER	The column header in the activity location file for transit accessibility. Default = <i>ACCESS</i> .
MODE_MAP_FILE – TRANSIMS	The mode file that defines a correspondence between TRANSIMS mode strings, i.e., <i>wcw</i> and an integer value that is used in a TRANSIMS activity file.
NET_DIRECTORY	The directory where the network files reside.
NET_NODE_TABLE	The network node table name.
NET_LINK_TABLE	The network link table name.
NET_ACTIVITY_LOCATION_TABLE	The network activity location table name.

Actgen is the program that generates simplified activities for each person in a located population file. Three activities are generated for each person (home, work, home). Times (start, end, and duration), work location, and mode choice are determined for each set of three activities. Several probability distribution parameters are used by the program (See Sections 2.6.1, 2.6.2, and 2.6.3).

Values for these parameters, as well as other program options, are set using TRANSIMS configuration file keys.

Usage: Actgen <configuration file>

Actgen uses the following configuration file key from the TRANSIMS configuration file in addition to the keys listed above.

Table 19: *Actgen* configuration keys.

Configuration Key	Description
ACT_FULL_OUTPUT	The name of the file where the generated activities will be written.

ActRegen is the program that regenerates leave home time, work locations, or mode choices for a traveler's activities. The IDs of the travelers to be regenerated and the type of information to be regenerated are read from a TRANSIMS feedback file of the following format.

<traveler id> <feedback command>

The valid feedback commands are shown in Table 20.

Table 20: *ActRegen* feedback commands.

Feedback Command	Description
L	Generates a new work location.
T<value in seconds>	Generates a new time to leave home for work. If <value> is given, add the given value to the new time generated. Value may less than zero.
M	Generates a new mode choice for the work and return to home activities.

Usage: ActRegen <configuration file>

ActRegen uses the following configuration file keys from the TRANSIMS configuration file in addition to the keys listed above.

Table 21: Additional *ActRegen* configuration keys.

Configuration Key	Description
ACT_PARTIAL_OUTPUT	The name of the file where the regenerated activities will be written.
ACTIVITY_FILE	The name of the file that contains the previous activity set.

3. ROUTE PLANNER MODULE

3.1 Overview

The purpose of the TRANSIMS Route Planner module is to generate routes for travelers. The routes between different locations are constrained by the network, which represents the metropolitan region being studied, and by preferences of individual travelers.

Information about each traveler's activities (contained in Section 2 of Volume 3—*Files*) is used to create *trip requests*. A trip request is defined by the origin and destination of a trip, the preferred starting time, and the *mode choice*. The mode choice defines the allowed modes of travel and their order, and is given in the form of a string of characters. The mapping between mode choices used in the activity file and strings used by the Route Planner is defined in the *Mode Preference File*. Additional information about vehicles that a particular traveler may use is contained in the *Vehicle File*.

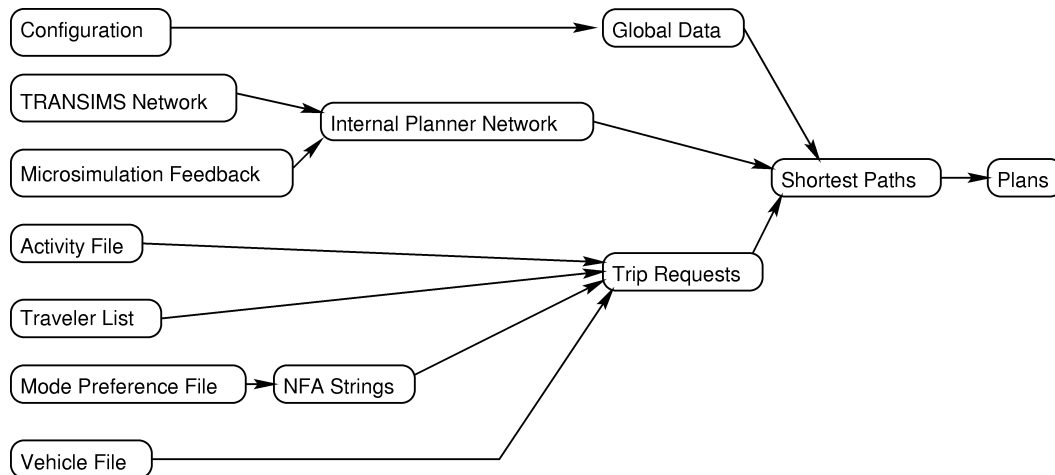


Figure 11: Data flow.

For efficiency purposes, the network used by the Route Planner is not the TRANSIMS network described in Section 3 of Volume 3—*Files*. The TRANSIMS network provides information about streets, intersections, signals, and transit in a road network. This information is used to construct the *internal Planner network*. The internal network is *time dependent*—that is, travel on a link may incur different delays at different moments in time. The information about delays on links is inferred from the Traffic Microsimulator output (*Feedback File*), which specifies the mean delays on each link over 15-minute intervals.

The core of the Route Planner is an implementation of Dijkstra's shortest-path algorithm with extensions for time-dependent delays and regular-language constrained paths. The internal network and trip requests are given to the path-finding routine, which creates routes and then outputs them in the form of *plans* (Section 4 of Volume 3—*Files*). For efficiency, the implementation of the Route Planner uses threads to allow parallel execution of several copies of the path-finding algorithm. Each thread uses the same copy of the network to create plans from different trip requests.

3.2 Algorithm

3.2.1 High-level Description

3.2.1.1 Dijkstra's Algorithm and Its Extensions

The internal network represents a weighted, directed graph. The nodes of the graph represent intersections and accessory locations (parking accessories, activity locations, transit stops), and the arcs (directed edges) represent travel possibilities between pairs of nodes. Internally, all links are unidirectional. Bidirectional TRANSIMS links are represented by two separate links in the Route Planner.

The algorithm underlying the TRANSIMS Route Planner is the classical Dijkstra's algorithm for finding shortest paths in a weighted directed graph. This algorithm can be viewed as breadth-first search of the graph, starting at the origin node and visiting the other nodes in the order of their (shortest-path) distance from the origin.

The actual algorithm used is a direct generalization of Dijkstra's algorithm. In fact, it may be viewed as Dijkstra's algorithm on a larger graph. In full generality, it is described by Barrett, Jacob, and Marathe.⁵ The same paper describes the extension for time-dependent delays. It turns out that a special case of time-dependency can be handled by Dijkstra's algorithm without special modifications (except, of course, the actual lookup of the correct delay values), and that this special case is enough for all practical purposes.

Consider a weighted graph whose links are labeled by possible modes of travel. A link may be traversed only in one of the modes whose labels appear on the link. We can constrain the shortest path by requiring that its label (the sequence of labels on the links in the path) belong to a regular language.

We represent the regular language by a directed graph, then construct a graph that is the cross-product of the network and the language. Finding a shortest path that belongs to the regular language then corresponds to finding a shortest path in this product network.

3.2.1.2 Generating Trip Requests from Activities

The activity, vehicle, and mode files are used to generate trip requests, which are then planned.

In the activity file, travelers' mode preferences are given by integers. Their meaning is defined by the mode file, which gives the correspondence between these integers and mode strings used by the Route Planner. After creating the internal network, the Route Planner reads the mode file and builds the mapping from mode preference integers to mode strings.

The Route Planner next looks at the configuration value ROUTER_HOUSEHOLD_FILE. If this value defines a file that can be opened, then only the travelers whose IDs are listed in this file will be routed. Otherwise, the Route Planner routes all the travelers in the activity file. All travelers in a single household are planned together because they may share transportation or activities.

⁵ C. Barrett, R. Jacob, and M. Marathe: "Models and Efficient Algorithms for Routing Problems in Time-dependent and Labeled Networks," Proc. 6th Scandinavian Workshop on Algorithm Theory, LNCS 1432.

The activities of each traveler are split into *legs*, which define either activities (*activity legs*), or travel (*transportation legs*). Each leg begins and ends at an activity location. The activity legs are written in the plan file. The transportation legs are planned. If a transportation leg is multimodal, it is planned as such and then further split up into unimodal sections, which are written into the plan file as separate legs of a *trip*. There is a slight inconsistency in terms: what is called a transportation leg in the context of the Activity File is actually a trip in the context of the plan file. However, this should not lead to much confusion.

If a trip to be planned uses a car, the vehicle file is examined to find the location of the car and the trip is split. The first mode string ends with the last symbol before “c”, and the destination of the first part of the trip is the parking accessory where the car is located. The second part of the trip starts there, with the mode “c”, and ends at the destination of the original trip. The two parts are planned separately and then written out in the plan file consecutively.

3.2.1.3 Parallelization

In order to increase the speed of the Route Planner, the work can be distributed among several processors when they are available. This is done using POSIX threads (pthreads). The Route Planner uses one master thread and one or more worker threads.

The master thread creates a queue of household trip requests and a queue of plans generated from the household trip requests. Each request contains the trip requests for each member of a particular household. The entire household is grouped together to allow for carpools. The master thread also constructs the internal planner network, which is accessed by each worker thread, and is responsible for writing the plans placed in the plan queue to the plan file.

Each worker thread removes a household trip request from the trip request queue and generates plans for the requests, as described in Section 3.2.2. The generated plans are then placed in the plan queue for output by the master thread.

3.2.2 Data Structures and Algorithm Implementation

3.2.2.1 Network Representation

The internal network can be visualized on a conceptual level, describing an abstract view of the network, but the implementation details diverge from this view in several points in order to allow more efficient algorithms. The conceptual representation is designed with the aim of providing an abstract representation of the network. We believe that such a representation will allow extending the network if additional modal attributes are given in the future. This level is also convenient for users and designers to view the system. The computer representation is more cryptic, but it is specifically designed with efficiency of the software in mind.

3.2.2.2 Layers: Concept

Conceptually, layers may be associated with modes of travel. In this view, there are three layers to the network: one represents the *street network*, consisting of all links between intersections, with added parking accessories. The second layer represents the *walk network* and consists of all streets that can be walked along and contains activity locations and other accessories. However, the parking accessories and transit stops that belong to the other two layers are only accessible from

activity locations. The final layer contains transit stops and *transit links*, which can only be traversed in transit (bus or light rail) modes.

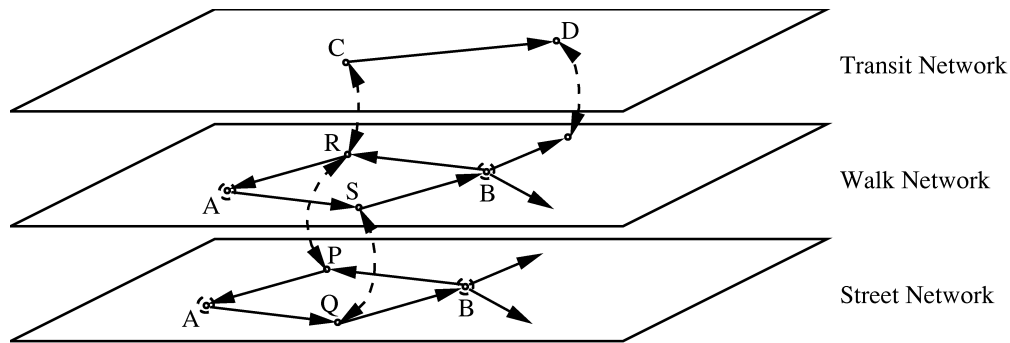


Figure 12: Conceptual diagram of the Planner Network. Parking accessories are in the Street Network, activity locations in the Walk Network, and transit stops in the Transit Networks.

In Figure 12 nodes A and B are street nodes and correspond to original TRANSIMS Network nodes. Nodes P and Q are parking accessories, while R and S are activity locations. Nodes C and D are transit stops. The links between layers are called process links.

Conceptually, nodes A and B appear in two different layers, even though these appearances correspond to the same TRANSIMS nodes. The reason for this is that even though we might be in the same geographic location whether in the street network or the walk network, we cannot change from the street to the walk network without visiting an activity location and using a process link.

3.2.2.3 Layers: Implementation

The actual representation of the network differs slightly from the conceptual description. The layers are numerical values associated with nodes. Information about nodes is stored in a single hash table, which allows access to a node by its external TRANSIMS ID and layer (a parking accessory can have the same TRANSIMS ID as a street node or an activity location), or by its hash index. Thus, nodes that represent parking accessories are for easier bookkeeping placed on layer 1, separate from TRANSIMS nodes that lie in layer 0. Activity locations are on layer 2, transit stops on layer 500, and nodes on transit line x on layer $500 + x$.

3.2.2.4 Accessories in the Internal Network

The TRANSIMS network defines the location of accessories (parking lots, activity locations, transit stops) by specifying the link on which they are located and the distance from one of the end nodes of this link. The accessories are represented by new nodes in the internal Route Planner network. Each link containing an accessory is split in two links, one from the start node of the link to the accessory node, and one from the accessory node to the end node of the link. The transformation is indicated in Figure 13.

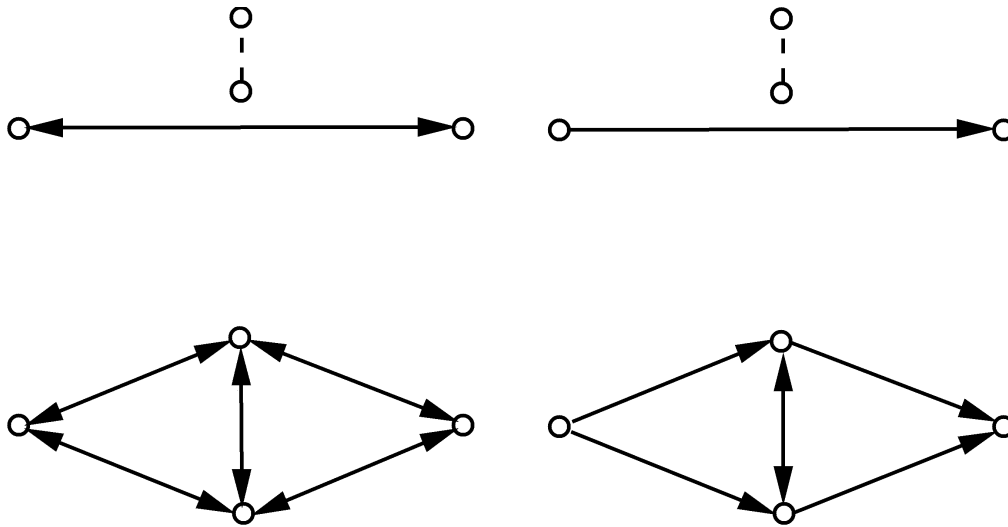


Figure 13: Splitting a link at accessory location. Top left/right: unidirectional/bidirectional link in TRANSIMS Network, with one parking accessory and one activity location. Bottom: the corresponding constructions in the Planner Network.

The information from the last two paragraphs allows us to describe the actual layered network representation used by the Route Planner. This is most clearly described by Figure 14.

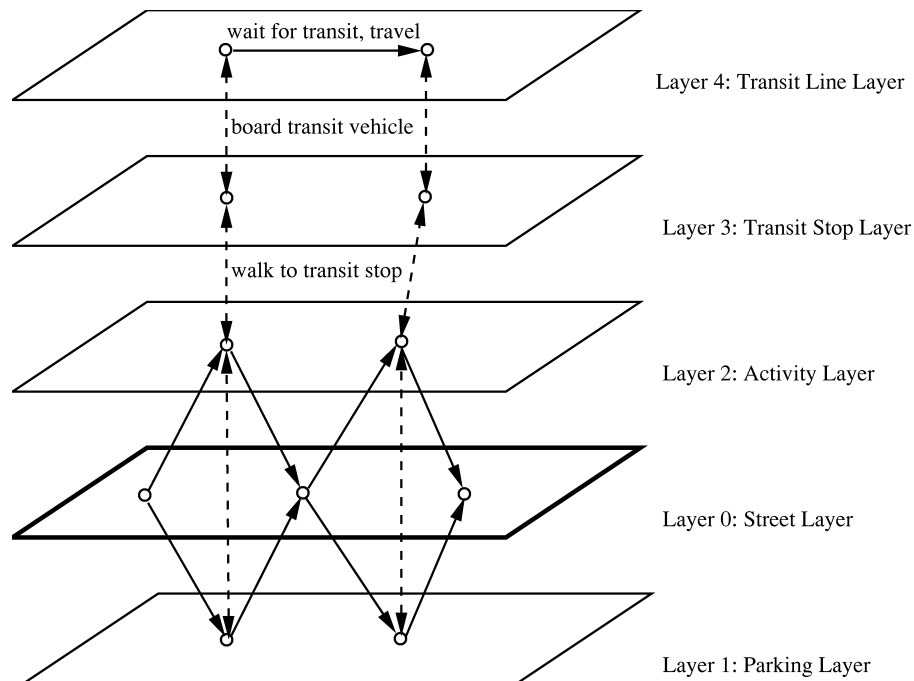


Figure 14: Layers of the Planner Network as represented by the actual data structures.

The differences between the implementation and the conceptual representation of the network are mainly in that some of the conceptually different entities in the network are represented by the same

elements of the network data structures, and vice versa. For instance, the TRANSIMS nodes are represented by a single entry in the implementation even though they may correspond to a node in the street network and another one in the walk network. On the other hand, the transit stops are implemented in two layers: first, each transit accessory has a node in the transit stop layer, and then each stop on each transit line has a node in the corresponding transit line layer. The transit accessories may be accessed by process links from activity locations. The link between the “external” transit stop and the node specific to the transit line may be used to represent the cost of taking the bus or train. The actual cost of waiting for the transit is coded in the delay function on the link between two nodes in transit line layer.

3.2.2.5 Graph Representation

The Route Planner assumes that all nodes in the network have low degree (in the Dallas car only studies, 8 was sufficient; for Portland EMME/2 and AllStreets networks in Portland, 16 will probably be enough).

With the introduction of bus stops, the degrees of some nodes become unacceptably high. To prevent this, the Route Planner creates additional copies of high-degree nodes when they become unavoidable, and inserts links with zero delay between the copy and the original. To handle such cases, another number is associated with each node, and is called *depth*. The depth of all “ordinary” nodes is 0. The first copy of each node is at depth 1, the second at depth 2, and so on.

The basic network is represented by an adjacency array: an array indexed by nodes. The elements of the array are arrays of node indices. For a node i , the entry at position $[i][j]$ is the index of its j -th neighbor. Since the maximum degree is low, the memory waste is not too big, and the use of arrays instead of lists allows fast access to the required links.

3.2.2.6 Assumptions about the Network made by the Route Planner

- All links to and from activity locations are explicitly given as process links. Just because an activity location is described by the TRANSIMS network as lying on a link is not enough to infer any connections to/from the activity.
- No accessory is located outside its link: $0 < \text{offset} < \text{length}$.
- No two parking accessories lie on the same link and have the same offsets.
- Each activity location is adjacent to a parking accessory. (This is not important if no trips are to be planned from the activity location that starts with “wc,” or to the activity, which ends with “cw”.)
- No link has length 0.

3.2.2.7 Product Construction

The problem with a practical implementation of the product construction described above is that the graph becomes too large. Therefore, we implement the product construction implicitly. With every node to be inserted in the priority queue, we associate the label of the link we used to arrive at this node. Whenever we examine the neighbors of an explored node to insert them in the queue,

we make two passes: first with the label (state) we are in now, then with the next label (state) allowed by the language.

This works because in the product graph, each link out of a node is either a link to a neighbor of the node in the same copy of the graph, or a link to another copy of the same node—either we keep the node the same and change the label (state), or we keep the label the same and change the node.)

3.2.2.8 Priority Queue

Usually, the set of all nodes visited so far is implicitly represented by writing down their shortest-path distance from the origin together with their immediate predecessor on the shortest path. This allows us to reconstruct the shortest path once we reach the destination node. We can say we have *explored* these nodes. The set of *fringe* nodes, the ones we can reach by edges from the explored nodes, is explicitly represented in the algorithm. Some of the fringe nodes may appear several times in this set because there may be links to the node from several explored nodes. With each element of this set, a number is associated—the length of the shortest path to the explored node, plus the length of the link to the fringe node. Thus, the shortest path to a fringe node is represented by the entry in which the fringe node appears with the smallest distance associated with it.

For an efficient implementation of Dijkstra’s algorithm, we thus need a data structure that will describe a weighted (multi)-set and allow us to efficiently insert elements and extract (and delete) the element with minimum weight. The data structure supporting these operations is in abstract terms called a *priority queue*.

Our implementation of the priority queue is a binary heap, represented in the memory by an array. We have tried using Fibonacci heaps, but the improvement was hardly visible in the running time (this was done during the Dallas case studies).

3.2.2.9 Classes

One basic class hierarchy represents the different levels of complexity of the network: `simple_nw` implements a weighted directed graph, `time_dep_nw` allows time-dependent delay functions, `accessory_nw` provides parking accessories and activity locations, and `busnet` implements a transit network together with label-dependent link delays, which allow multimodal routing. All of these could theoretically be used as graphs on which we plan trips—but memory requirements may preclude using more than one type of network at any one time.

Table 22: Network classes.

Class	Functionality
<code>simple_nw</code>	static layered network (weighted directed graph)
<code>time_dep_nw</code>	time-dependent link costs
<code>accessory_nw</code>	parking and activity accessories, splitting links
<code>busnet</code>	transit layer, delay functions for transit links

Busnet is thus used as the internal network for the Route Planner, and a single copy of busnet is constructed and used by all threads during the execution of the Route Planner.

In order to allow mode-specific optimization, there is a separate class hierarchy that implements operations needed by the Dijkstra's algorithm (lookup of degrees, neighbors, and link delays). The base class is called `Routing_Network`, and is inherited by `NFA_Net`, `Car_Net` and `Walk_Net`. These three classes also contain data members describing a trip request such as `origin`, `destination`, `nfa` (mode string).

Table 23: Routing classes.

Class	Functionality
<code>router_net</code>	base class for other routing classes; some common functions
<code>NFA_Net</code> <code>Car_Net</code> <code>Walk_Net</code>	functions needed by Dijkstra's algorithm, optimized for the specific mode

A general implementation of Dijkstra's algorithm is provided by `path_finder`, a template class that can be used with different routing networks and different priority queues. Currently, only one priority queue implementation is available.

3.2.3 Optimization

3.2.3.1 Routing Classes

The multimodal mechanism is too general for some applications since it provides more than may be needed, at a higher cost in execution time. So we turn to special cases of interest that can be handled in more efficient ways.

To avoid overhead associated with each link delay inquiry, we use separate routers for different kinds of trips. The different routers will call different delay and neighbor functions, and will be able to see past irrelevant portions of the network without any overhead. The above-mentioned routing classes `NFA_Net`, `Car_Net`, and `Walk_Net` provide these operations, and the path-finding algorithm invoked for a particular trip uses the operations appropriate for the mode preference associated with the trip. Thus, all overhead is reduced to a switch statement executed once per trip, to code duplication for each different optimized mode preference, and to memory in the unused classes (two at each moment).

For example, the first optimized classes we implemented are called `NFA_Net`, `Car_Net`, and `Walk_Net`. Each of them contains a reference to the `mininfanet` object constructed from the TRANSIMS network. They contain functions necessary to run Dijkstra's algorithm: `first_neighbor` and `last_neighbor`, which provide the bounds for a loop through all neighbors of a given node, and `get_next`, which returns the next neighbor and the time needed to traverse the link from the current node to the next neighbor.

`Car_Net` is used with "wcw" (all plans now begin and end at activity locations, so at beginning and end the traveler must walk from a parking accessory to an activity location). Currently, we

assume that the parking accessories are adjacent to the origin and destination activity locations, and we really only plan the car leg of the trip. Walk_Net is used with “w”.

The Route Planner constructs Path_Finder<Car_Net>, Path_Finder<Walk_Net> and Path_Finder<NFA_Net>, and given a trip, uses the correct version to find a path.

The class NFA_Net is the most general multimodal mechanism available, but also the slowest. On the other hand, Car_Net and Walk_Net are very fast, working only on the relevant part of the network (thanks to the functions first_neighbor and last_neighbor, the links not labeled as car-links or walk-links, respectively, need not even be considered during the search). These classes can also use knowledge about the structure of the particular network we work with, such as when planning a car trip, if we know the origin and destination parking accessories, we can ignore the parking accessories in between. This effectively reduces the size of the network and the length (in the number of links) of paths in the network, improving the running time. If structural information like this is not available, the best we can do is use the general multimodal router.

3.2.3.2 Allowed Mode Preferences and Extensions

The current formulation allows constraints described by *linear* languages—those of the form $a_1^b a_2^b \dots a_k^b$ where b may stand for 1 (one occurrence of the symbol), + (one or more), or * (zero or more). The letters used are “w” (walk), “c” (car), “b” (bus), and “t” (transit). Process links are designed to be walked on, and a fixed cost is associated with traversing a process link. Transit means bus or walk.

Any linear expression consisting of the above symbols can be planned. Some, of course are meaningless in that they will never produce paths (e.g., “cb”—because a process link must be traversed between a parking accessory and a bus stop). Some others may not produce exactly what is expected: “b” restricts the plan to boarding only one bus since the process links between bus line nodes and bus stops are traversed in walk mode—we probably need to add a letter (say “e” as enter, exit) for the links between bus stops and bus line nodes, and then we can make “t” mean “e” or “b”.

In the general mechanism (when NFA_Net is used), all that the Route Planner needs is to be able to determine the delay incurred by traversing a link in a certain mode. It would be easy to include an array listing link delays indexed by link IDs and modes, if new modes of travel are added.

However, if there is more information about a specific type of mode preference, it is possible to optimize the planning by writing a class parallel to, e.g., Car_Net. For example, if we must distinguish between local streets and highways, the following could provide a solution that can be extended without many problems later (this need not mean new modes of travel, one for each type of street; the solution we propose effectively removes some links from the network by instructing the shortest-path code not to explore them)

3.2.3.3 Heap Details

HA heap is a data structure that stores elements of a given set and allows access according to their values. The elements are stored so that the element with minimum value can be extracted quickly. The operations supported are remove_min, which removes the element with minimum value, and insert_update, which inserts a new element into the heap. We use the standard implementation of a binary heap as an array.

3.2.3.4 Heuristics

Routing classes `Car_Net` and `Walk_Net` can also use the *Sedgewick-Vitter heuristic* for Euclidean graphs. The heuristic allows finding almost optimal shortest paths between nodes in a Euclidean graph. A parameter, called `overdo` allows for a trade-off between the running time and optimality of the paths found. Of course, the internal network is not strictly Euclidean, since only certain nodes may be reached from each node (the graph is not complete), but we have found that the paths produced with moderate values, such as `overdo = 3` look quite realistic and bring a considerable improvement in running time.

However, if this heuristic is used, the plans will be less sensitive to the feedback. The larger the value of `overdo`, the larger congestion will be tolerated by the Route Planner before alternative routes are taken.

3.3 Usage

Make sure that all files listed in the configuration file and mentioned below exist. Then, run the Route Planner with

```
Router config
```

where *Router* is the name of the Route Planner executable, and *config* is the name of the configuration file.

3.3.1 Configuration Values Used by the Route Planner

3.3.1.1 Files

Table 24: Files configuration values.

Configuration Key	Description
ACTIVITY_FILE	Path to a TRANSIMS activity file. Required.
VEHICLE_FILE	Path to a TRANSIMS vehicle file. Required.
MODE_MAP_FILE	Path to a mode file. Required.
ROUTER_HOUSEHOLD_FILE	Path to a file containing a list of integer IDs for travelers to be planned. Optional.
PLAN_FILE	Name of the file where plans should be written. (Overwrites an existing file.) Required.

Although configuration parameters exist for them, the Route Planner also needs an activity index file and a vehicle index file. They can be created by calling utilities *IndexActivityFile* and *IndexVehFile* with full pathnames of the activity file and vehicle file as arguments.

3.3.1.2 Threads

Table 25: Threads configuration values.

Configuration Key	Description
ROUTER_NUMBER_THREADS	Positive integer. Number of routing threads to be used. Default: 1. Optional.

3.3.1.3 Network, Data Adjustments, Planning Heuristics.

All parameters are optional.

Table 26: Network, data adjustments, and planning heuristics files.

Configuration Key	Description
ROUTER_OVERDO	Nonnegative float. If set to 0, no adjustment is made to the distance estimates. If positive, the search for the shortest path to the origin will be biased in the direction of a straight line to the destination. This will produce nonoptimal paths. The paths will still be reasonable, but the heuristic may cause relatively small congestion on links to be ignored, and this can break the iterative relaxation mechanism. Default: 1.0.
ROUTER_CORR	Float, between 0 and 1. Some of the networks we received had errors. In particular, the reported length of a link was often much smaller than the actual Euclidean distance between the two endpoints of the link. This may cause problems when the Sedgewick-Vitter heuristic is used. The Route Planner will change the reported length of a link to be equal to its Euclidean length whenever the ratio of the two is less than ROUTER_CORR. Default: 0.0.
ROUTER_ZERO_BACKD	Integer, 0 or 1. Default: 0.

3.3.2 Data Structure Parameters and Verbosity

All parameters are optional.

Table 27: Data structure parameters.

Configuration Key	Description
ROUTER_MAXNFASIZE	Positive integer. Maximum length of a mode string to be used. Influences the size of priority queue and shortest path tree. (The number of entries allocated roughly equals the size of the network times two times MAXNFASIZE.) Default: 5.
ROUTER_MAX_DEGREE	Positive integer. Maximum number of links that may leave a node. Should be at least double the maximum degree of the TRANSIMS Network because walk links and street links are separate. Default: 8.
ROUTER_INTERNAL_PLAN_SIZE	Positive integer. Should be enough to accommodate the length (in number of nodes) of the shortest path between any two nodes in the network (and may need to be quite large when multimodal plans are used). Default: 400.
ROUTER_HASH_OVERSIZE_FACTOR	Float, greater than 1. The factor by which the size of allocated hash tables is greater than the estimated number of links and nodes. Default: 1.5.
ROUTER_VERBOSE	Nonnegative integer. Values above 2 generate a lot of diagnostic output, and high values such as 10 allow inspection of almost everything the Route Planner does, from creation of the network to step-by-step expansion of Dijkstra's algorithm and the shortest path search. May also be turned off (which speeds up the program) by undefining VERBOSE during compilation.

3.3.2.1 Assumptions the Route Planner Makes on the Network

- 1) Each activity location that is used as the origin or destination for car trips should be adjacent to a parking accessory. Since the trips of the form "wcw" (planned by *Car_Net*) are split into "w" and "cw" this assumption may not be necessary for origins of trips, only for destinations. The reason is that the car trip is planned as a unimodal trip of the form "c" and the last walk leg is planned without calling Dijkstra's algorithm, just by finding a parking accessory adjacent to the destination activity. The Route Planner prints an error message if this assumption is not satisfied, and exits.
- 2) No (parking or activity) accessory is located off the end of its link. The location of an accessory on a link in TRANSIMS is specified by its distance from the endnode of the link, the value called *offset*. The Route Planner assumes that no offset is negative and that every offset is less than the length of the corresponding link. If this assumption is not satisfied, the Route Planner prints warnings. It proceeds in planning, but its behavior (especially with respect to calculating distances) is not defined.
- 3) Connectivity. If there is no path between the origin and destination nodes that satisfies modal constraints, the Route Planner prints a warning, *No Path found for traveler xxx*, and proceeds in planning. The corresponding transportation leg is not written in the plan file.

3.4 Troubleshooting

Problem 3.4.1:

Using multiple threads (ROUTER_THREADS set to more than 1) crashes the Route Planner on a Solaris system using an executable compiled with g++.

Solution 3.4.1:

This is caused by a problem in the interaction of g++ and pthreads under Solaris. It does not happen on a Linux system, or on Solaris when using the SunPro compiler.

Problem 3.4.2:

Route Planner crashes with a segmentation fault.

Solution 3.4.2:

Make sure that all filenames are correctly defined, and that indexes for activities and vehicles are up-to-date.

Problem 3.4.3:

Iteration between the Route Planner and the Traffic Microsimulator produces feedback information but vehicles keep using the same congested routes.

Solution 3.4.3:

Reduce the value of ROUTE_OVERDO. If this value is high, the congestion will have less influence on the paths found because the search for shortest paths will be biased toward geometrically shortest paths with little regard to their actual cost. With `overdo` set to 0, no bias is used and actual shortest paths are found.

4. TRAFFIC MICROSIMULATOR MODULE

4.1 Overview

The TRANSIMS Traffic Microsimulator module simulates the movement and interactions of travelers in the transportation system of a metropolitan region. Using a trip plan provided by the Route Planner, each traveler attempts to execute the plan on the transportation system. The combined traveler interactions produce emergent behaviors such as traffic congestion.

The Traffic Microsimulator simulates intermodal travel plans, multiple travelers per vehicle, multiple trips per traveler, and vehicles with different operating characteristics. Roadway transportation was chosen for the initial emphasis because of its high use, complexity, and importance to air quality. The roadway network includes freeways, highways, streets, ramps, turn pocket lanes, and intersections (signalized and unsignalized). Vehicle drivers executing trip plans accelerate, decelerate, turn, change lanes, pass, and respond to other vehicles, signs, and signals.

The Traffic Microsimulator uses a cellular automata approach to provide the computational speed necessary to simulate an entire region at the individual traveler level. The cellular automata technique provides a means to simulate large numbers of vehicles and maintain a fast execution speed. Each link in the transportation network is divided into a finite number of cells. At each timestep of the simulation, each cell is examined for a vehicle occupant. If a vehicle is present in the cell, the vehicle may be advanced to another cell using a simple rule set. Increasing the fidelity by decreasing the cell size, adding vehicle attributes, and expanding the rule set results in slower computational speed. The fidelity and performance limits of the cellular automata microsimulation are evaluated to establish the computational detail required to support the fidelity necessary to meet analysis requirements.

The sheer number of travelers and the level of detail in the microsimulation dictates the use of multiple CPUs where available. The description in the “Algorithm” section includes an explanation of the information flows and scheduling required to support parallel computing. For simplicity, we first present an overview of the simulation as it would be carried out on a single CPU.

First, a representation of the transportation network is read in. This representation is very similar to a detailed street map, including numbers of lanes, turn pockets, merging lanes, turn signals, etc. Vehicles traveling along streets in the road network are simulated in detail. In addition to the streets, there are several kinds of accessories, which act like buffers for travelers who are not in a vehicle traveling on a street.

Next, the type and initial location of every vehicle is read in. Then traveler’s plans are read in as needed. The travelers are placed on the network and allowed to move from their origin to destination. For non-simulated modes, this movement is simple—a traveler is removed from the buffer in one accessory and placed in the buffer on another, with a new departure time reflecting the estimated duration of the trip. Vehicles are moved from one grid cell to another using the CA approach described above with modifications to support lane-changing and plan-following until they reach the end of a grid. There they wait for an acceptable gap in traffic or for protection from a signal before moving through the intersection onto the next grid. This continues until each vehicle reaches its destination, where it is removed from the grid.

Events requested by the user, as well as evolution data, are collected by the output system (*q.v.*) at each step of the simulation.

4.2 Algorithm

The procedures invoked during each simulation timestep can be placed into one of five broad categories:

- 1) placing travelers and vehicles
- 2) updating the location of each traveler and vehicle
- 3) preparing for a timestep
- 4) cleaning up after a timestep
- 5) supporting parallel computation

Each of these is discussed, in this order, in this section.

4.2.1 Placing Travelers and Vehicles

Figure 15 shows the processes and data structure involved in loading travelers and vehicles into the Traffic Microsimulator.

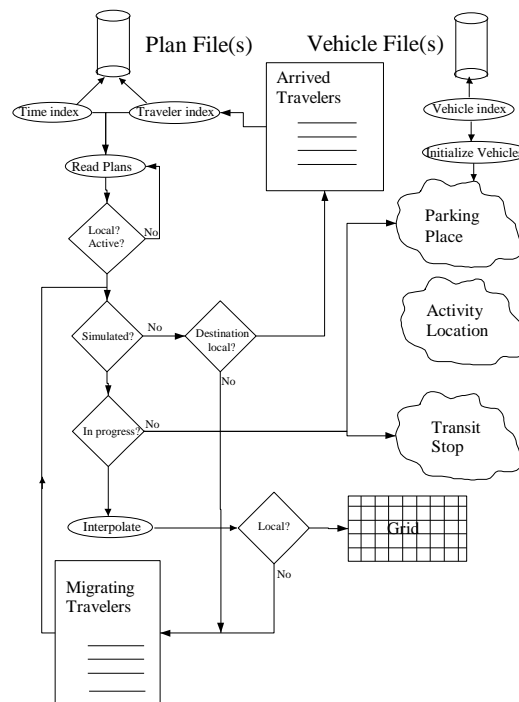


Figure 15: A flow chart of the processes and data structures involved in getting vehicles and travelers into the simulation.

The vehicle and plan files contain all of the information required by the simulation in addition to the network files (not shown in Figure 15). Both are accessed through an index, which will be generated from the appropriate file if it does not already exist. Note that an index can refer to more than one data file. Also, there may be a separate index for each processor (if the configuration key CA_USE_PARTITIONED_ROUTE_FILES is set).

A list of vehicle IDs located at each parking accessory is created in the *Initialize Vehicles* phase. The corresponding vehicle object is not created until it is used, to reduce memory requirements.

Traveler plans (or, more correctly, legs of a plan) are read using the index sorted by expected departure time until all plans departing before or on the current simulation step have been read. In addition, the IDs of *hibernating* travelers (those who have already executed one leg of their plan and are waiting to depart on another) are popped off the queue of *Arrived Travelers*. Each hibernating traveler carries along a minimal required information set consisting of traveler ID, current trip and leg ID, and a set of state flags used in maintaining states required by the output system. Other non-essential information is deleted from memory while a traveler hibernates to minimize memory requirements. The *Read Plans* process uses an index into the plan file which is sorted by traveler ID to find the next leg for each of the arrived travelers.

Each plan must pass two tests before the traveler is placed onto the transportation network:

- 1) It must be *local*: its origin must be an accessory that is a part of the network controlled by the CPU.
- 2) It must be *active*: its expected arrival time must be after the simulation start time, and its departure time must be before simulation end time. Simulation start and end times are defined by the configuration keys CA_SIM_START_HOUR, CA_SIM_START_MINUTE, CA_SIM_START_SECOND, and CA_SIM_STEPS.

Next, if the plan calls for a non-simulated mode of travel (activity, walk, or bicycle) and the destination is local, the process attempts to place the traveler in the *Arrived Traveler* queue with a departure time specified by the plan. For example, the plan might say to use the later of a 10-minute duration or 8:10 a.m. The simulation will add ten minutes to the current simulation time, compare it to 8:10 a.m. and place the traveler into the queue with a departure time equal to the later of the two. If the destination is not local, the traveler must migrate to another CPU, where it will be placed into the Arrived Traveler queue for that CPU.

If the traveler is using a simulated mode of transportation (anything involving a vehicle, as passenger or driver), and the plan is not in progress (i.e., its departure and arrival times do not straddle simulation start time), the traveler is placed in a queue in his/her origin accessory. This could be either a transit stop or a parking accessory. Currently, no travelers will be placed in activity locations because there is no path to or from these using simulated transportation modes.

It is desirable for the simulation to reach normal traffic flow conditions as rapidly as possible. This is facilitated by placing vehicles on the roadway when the simulation is initialized according to where their driver's plans predict they will be at the simulation starting time. The geometric length of a plan is estimated, and a link is chosen by interpolating along the path according to the duration of the leg as estimated by the Route Planner. The length is difficult to determine if the plan is not wholly contained within the part of the network local to the CPU, so this process is not guaranteed to produce the same initial condition because the number of CPUs varies. If the interpolation process determines that a traveler should be placed on a non-local section of the

network, it will add the traveler to the list of migrating travelers. Otherwise, the cell position and lane are chosen randomly. If the chosen cell is already occupied, the grid is searched upstream for an available cell. If all cells upstream are occupied, the grid is searched downstream for an unoccupied cell. If all cells on the link are occupied, a warning message is printed and the vehicle is deleted. No attempt is made to find an available cell on an adjacent link. Since interactions between vehicles are not taken into account, this procedure does not produce the same distribution of traffic that would be found by starting earlier and letting the simulation evolve to the same time.

4.2.2 Updating traveler locations

After reading in and placing travelers, the simulation executes their plans one step at a time. Each step involves several substeps in the order given below and in Figure 16. The major data structures involved and their interactions during a timestep are sketched in Figure 17.

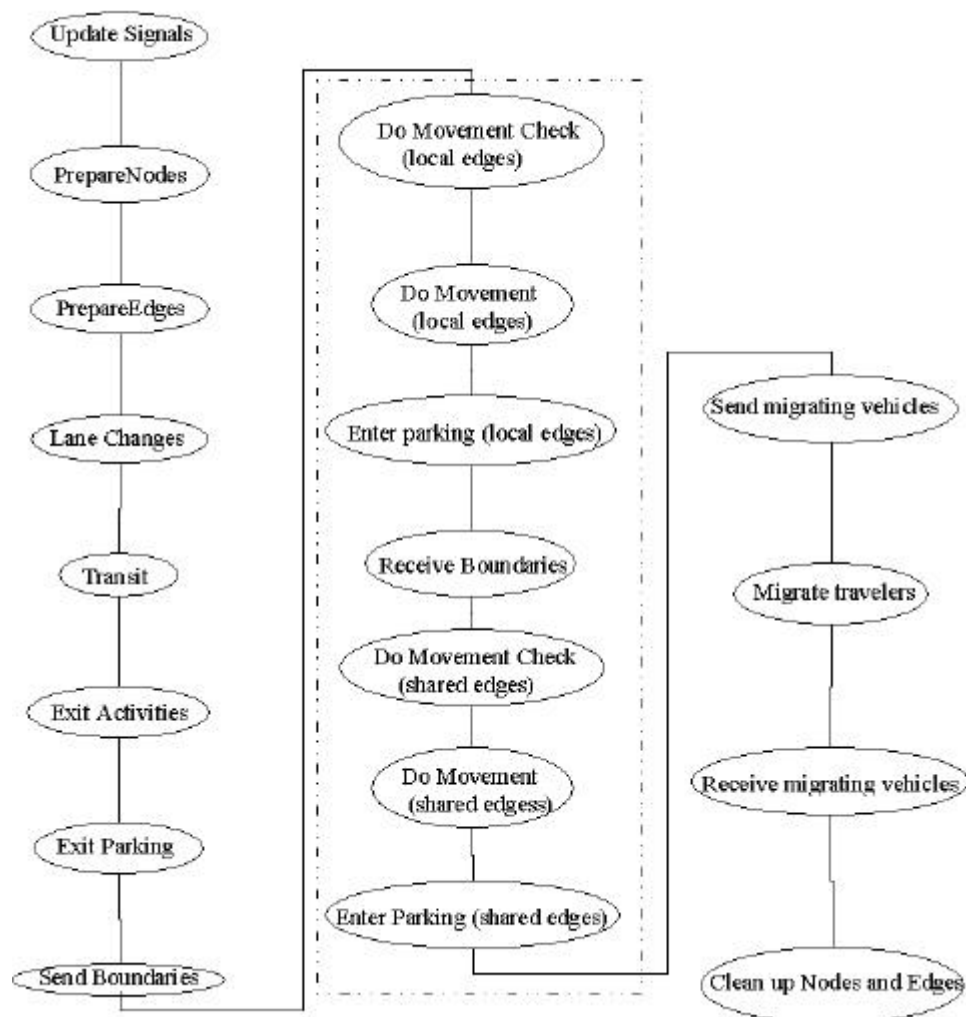


Figure 16: Order of execution of processes in each timestep.

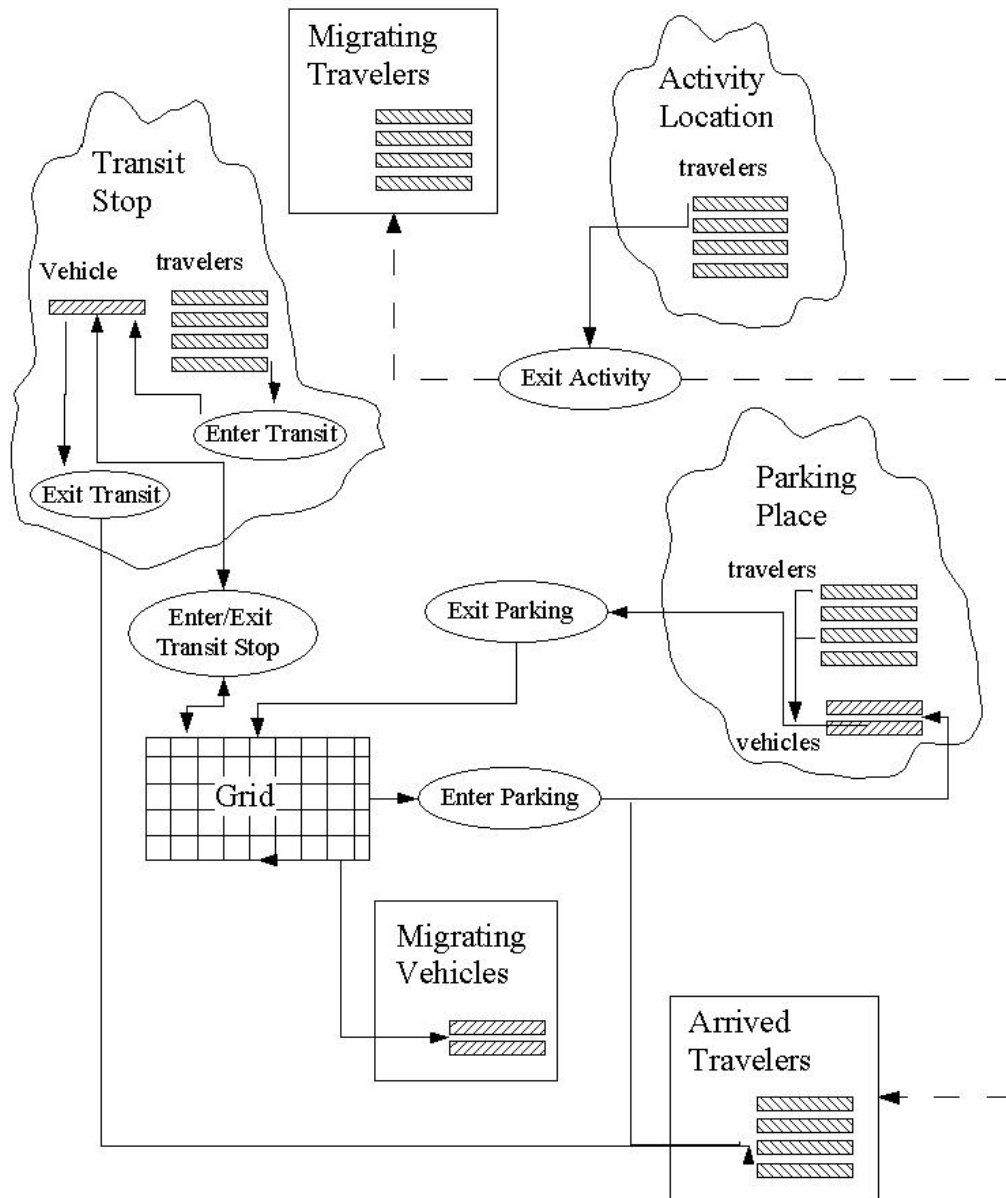


Figure 17: Interactions among objects during a timestep.

4.2.2.1 Traffic Dynamics

Traffic dynamics in the Traffic Microsimulator are produced by interactions of individual vehicles on the transportation network. The position of vehicles on the roadway is determined by applying a rule set that governs movement and lane changes. This rule set must be as simple as possible in order to maintain the computational speed necessary to update positions of the large number of vehicles that could be present in a regional traffic microsimulation. The rule set imposes a no-collision strategy on the vehicles. Vehicle interactions based on the rule set combine to produce emergent driver behavior. Traffic dynamics require that, for any vehicle v at time t , all position change calculations must be based on other vehicle positions at time t , not at time $t + 1$.

4.2.2.2 Definitions

Timestep One microsimulation update cycle in which all movement and lane changes are executed for each vehicle. Each timestep typically represents approximately one second of simulation time.

Grid Division of the link into cells forming a grid (Figure 18). The Traffic Microsimulator uses a separate grid for each lane on the roadway. Each cell is 7.5 meters long. Vehicles may occupy more than one cell of a grid. In particular, the length of a transit vehicle (in cells) may be set using the configuration key CA_BUS_LENGTH.

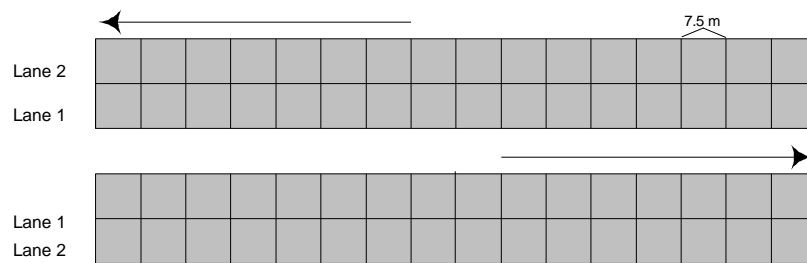


Figure 18: Link with grid cells.

Gap Number of empty cells between this vehicle and the next vehicle on the grid (Figure 19). If this is the first vehicle on the grid, gap is the number of empty cells between this vehicle and the end of the grid.

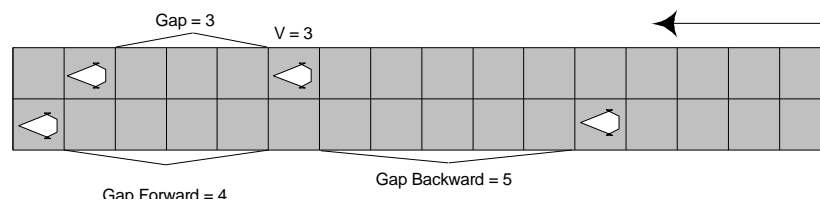


Figure 19: Gaps.

V Speed of the vehicle in cells/timestep.

V_{Max} Speed limit on the link in cells/timestep.

$V_{GlobalMax}$ Maximum speed on any link in cells/timestep.

P_D Deceleration probability. Probability that a vehicle will decelerate during a timestep. Set using the configuration key CA_DECELERATION_PROBABILITY.

P_L Lane changing probability. Probability that a vehicle will change lanes during a timestep for reasons other than plan following. Set using the configuration key CA_LANE_CHANGE_PROBABILITY.

D_{PF}	Distance from the intersection where a vehicle starts to consider changing lanes in order to follow its plan. Set using the configuration key CA_PLAN_FOLLOWING_CELLS.
N_{rand}	Random number between 0.0 and 1.0. The random number sequence can be controlled using the configuration keys CA_RANDOM_SEED1, CA_RANDOM_SEED2, and CA_RANDOM_SEED3, which are all combined into a single seed.
Pocket Lane	A pocket lane is a lane that does not extend the whole length of a link. A turn pocket starts mid-link and ends at the intersection, whereas a merge pocket starts at the intersection and ends mid-link. A link may have a left or right turn pocket to facilitate turns at an intersection. Merge pockets facilitate entry onto a link.

4.2.2.3 Lane Changes

First, we examine every vehicle to see whether it will change lanes on this timestep. It is necessary for lane change and movement to occur on the same timestep to produce realistic traffic dynamics. Left and right lane changes are made on alternating timesteps to prevent collisions and to ensure that gap calculations are based on vehicle positions at time t , not $t+1$. Multilane roadways are processed from left to right when making left lane changes and from right to left when making right lane changes. Vehicles are not allowed to change into a lane if it would violate lane use or HOV restrictions.

Vehicles will change lanes for two reasons:

- 1) to pass a slower vehicle in the current lane
- 2) to make turns at intersections in order to follow its plan

The decision to make a lane change in order to pass a slower vehicle is based on the gap on the current lane, the gap backward on the new lane, and the gap forward on the new lane.

A vehicle that needs to make a turn at the next intersection in order to follow its plan will start to consider a lane change when it is within a set distance from the intersection. As the vehicle approaches the intersection, the urgency to change into a lane that is appropriate for plan following increases linearly as the vehicle approaches the intersection. Any vehicles that fail to make the required lane changes for plan following are marked as off-plan..

4.2.2.3.1 Passing Lane Change

Passing lane changes are based on three gap calculations (Figure 18):

- 1) Gap in the current lane (G_c)
- 2) Gap forward in the new lane (G_f)
- 3) Gap backward in the new lane (G_b)

These gaps are used to set the weight values (Figure 19) that are in the calculations to determine if a lane change will be made. The calculated gaps and

the potential speed of the vehicle in the current timestep are used to determine whether a vehicle will make a lane change for passing.

Table 28: Weight values.

Parameter	Description	Equation
Weight 1	An integer value based on the gap in the current lane, the potential speed of the vehicle in this timestep, and the gap forward in the new lane.	Weight 1 = ((V+1 > G _c) AND (G _f > G _c))
Weight 2	An integer value based on the gap forward in the new lane and the speed of the vehicle.	Weight 2 = V - G _f
Weight 3	An integer value based on the gap backward in the new lane and the maximum speed of a vehicle in the simulation.	Weight 3 = V _{GlobalMax} - G _b
Weight 4	An integer value based on the distance from the intersection (D _i) and point on the link where a vehicle starts to consider lane changes to follow its plan (D _{PF}).	Weight 4 = V _{GlobalMax} - (D _i /nD _{PF}) (V _{GlobalMax} - 1)

A vehicle will make a lane change for passing if the following three conditions are satisfied:

- 1) Weight 1 > 0
- 2) Weight 1 > Weight 2
- 3) Weight 1 > Weight 3

4.2.2.3.2 Plan Following Lane Change

Acceptable approach lanes that allow a vehicle to transition to the next link in its plan are determined when a vehicle enters a link. Lane changes for plan following are introduced into the lane change calculations when a vehicle is within a set distance from an intersection (D_{PF}). The bias to make a plan following lane change increases as the vehicle nears the intersection. If the vehicle is already in an acceptable approach lane, the vehicle is biased to stay in the correct lane and ignore lane changes to pass slower vehicles (i.e., lane changes based on gaps).

Plan following lane changes are controlled by introduction of an additional parameter to the lane change calculations. The parameter, Weight 4, is initially set to zero.

If the vehicle is within the D_{PF} and is not in an acceptable approach lane, Weight 4 is set based on the distance between the vehicle and the intersection (D_i). Weight 4 increases as the vehicle moves nearer to the intersection.

Since only one type of lane change is made during a timestep, the type of lane change needed (left/right) must be the same as the type of lane change (left/right) that is calculated during this timestep.

If a vehicle is already in an acceptable approach lane and is within the D_{PF} , Weight 4 = -1, which will prevent any lane changes based on gaps (passing lane changes).

It is possible for a vehicle to have more than one approach lane that is acceptable for plan following. If the vehicle is in an acceptable lane and the new lane (left/right) is also an acceptable approach lane, Weight 4 = 0, which allows lane changes based on gaps.

The final calculation to determine the lane change calculates Weight 1 using Weight 4.

$$\text{Weight 1} = ((V+1 > G_c) \text{ AND } (G_f > G_c)) + \text{Weight 4}$$

Weights 2 and 3 are calculated as indicated in the passing lane changes above.

A vehicle will make a lane change if the following three conditions are satisfied:

- 1) Weight 1 > 0
- 2) Weight 1 > Weight 2
- 3) Weight 1 > Weight 3

4.2.2.3.3 Special Cases

4.2.2.3.3.1 Mass Transit

Mass transit vehicles must not become off-plan and must have priority in making lane changes, so they are handled separately by this algorithm. Each transit vehicle will enter a transit stop if it is not full and there is a queue of people waiting at the stop or if any passenger wishes to get off at the stop. The vehicle will either be left occupying the grid cells or taken off the grid entirely, depending on the style of transit stop (e.g., 'YARD', 'STATION', or 'STOP'). If it is left on the grid, it will attempt to get into an appropriate lane. The vehicle's speed constraint is set to 0 while it is in the stop.

4.2.2.3.3.2 Merge Lanes

Vehicles in merge lanes are forced to make lane changes in the same direction as the merge direction. In some cases, a lane can have a merge pocket and a turn pocket further down the lane toward the intersection. In this case, vehicles are prohibited from entering the lane until they are past the end point of the merge pocket.

4.2.2.3.3.3 Turn Pocket Lanes

Speed restrictions are imposed on a vehicle attempting to enter a turn pocket lane from the adjacent lane. These restrictions prevent movement of the vehicle past the start of the turn pocket, which causes the vehicles to queue on the adjacent lane until a lane change into the turn pocket lane is possible.

In Figure 20, the vehicle in Lane 2 needs to make a left turn at the next intersection. The left turn pocket (Lane 1) has no vacant cells. At time t , the vehicle's speed is 3, which will move the vehicle past the start of the turn pocket. The vehicle's speed is constrained to 2 (the distance from the vehicle's current position at time t and the start of the turn pocket). At time $t+1$, the vehicle has moved down Lane 2 to the starting cell of the turn pocket. A lane change into the turn pocket is not possible because all of the cells are occupied by other vehicles. The vehicle is prevented from traveling further down Lane 2 by constraining the speed to 0. At time $t+2$, the vehicle remains on

Lane 2 with speed 0. The vehicle's speed will remain constrained to 0 until a lane change into the turn pocket is possible.

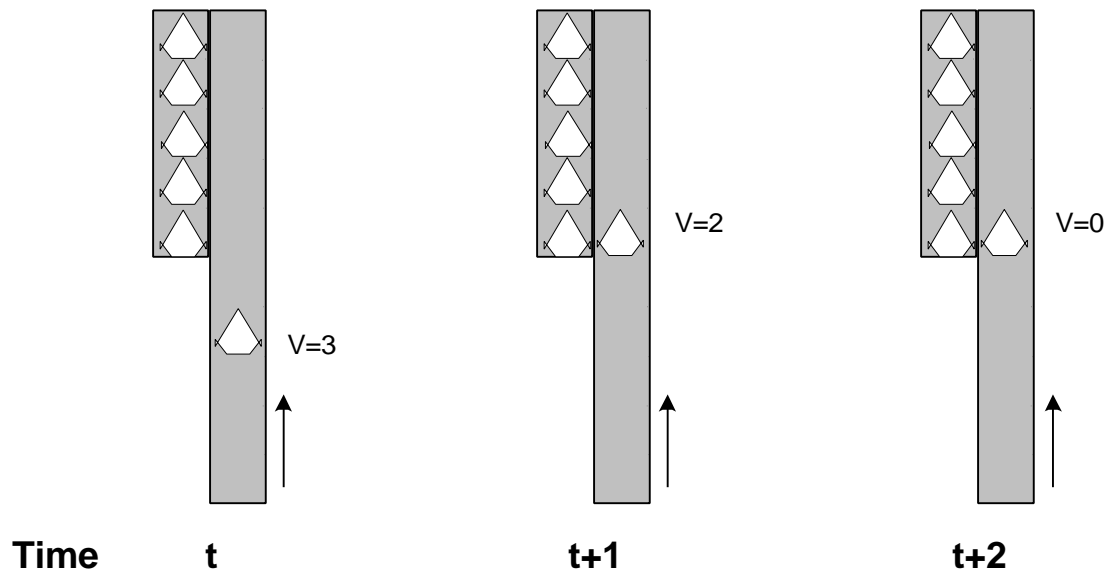


Figure 20: Turn pocket lanes.

4.2.2.3.3.4 Look Ahead across Links

If acceptable, approach lanes are determined considering only the connectivity to the next link; some vehicles may be unable to make the required lane changes into acceptable approach lanes on short multilane links with multiple lane connectivity at the intersections. Looking ahead across links increases the time that a vehicle has to make a plan following lane change.

The acceptable approach lanes are determined based on a plan look ahead distance. The distance is used to determine how many links in the plan will be considered when determining the approach lanes on the current link. Plan look ahead distance is described in the next section. A distance of 262.5 meters (35 grid cells) is the default value. A value of 0.0 means that approach lanes are determined by considering the next link only.

4.2.2.4 Transit

Next we examine each transit stop. The transit stop object contains a pointer to a vehicle (implying that the capacity of stops is 1) and a queue of travelers.

If there is a transit vehicle currently servicing the stop, and it has been there for at least the number of timesteps specified by the configuration key `CA_TRANSIT_INITIAL_WAIT`, travelers are allowed to enter and exit the vehicle. Entry and exit can happen simultaneously, but the mean rate at which travelers enter and exit is set by the configuration keys `CA_ENTER_TRANSIT_DELAY` and `CA_EXIT_TRANSIT_DELAY`, respectively.

Travelers are popped off the traveler queue until we reach either the maximum number of travelers who can board in a single timestep or a traveler whose next departure time is later than the current simulation time. If the traveler's plan calls for him/her to take the route that this vehicle is

servicing, and the number of passengers already aboard does not exceed the value specified by the configuration key CA_BUS_CAPACITY, he/she will enter the vehicle.

Travelers leaving the vehicle have completed a leg of their plan and are placed in the *Arrived Travelers* list to trigger the *Read Plans* process to find the next leg of their plans. If all of the passengers entering and exiting at this stop have been taken care of, the vehicle is placed back on the grid (if necessary) and its speed constraint removed.

4.2.2.5 Exit Activities

Activity Locations contain a queue of travelers, with their associated plans. During this process, each traveler scheduled to depart at this timestep is popped off the queue. The traveler is then treated as if his/her plan had just been read by *Read Plans*. Since there is currently no way to leave an activity location using a microsimulated transportation mode, the effect of this step is to place a hibernating traveler on the *Arrived Travelers* queue if the destination is local or on the *Migrating Travelers* list if it is not.

4.2.2.6 Exit from Parking Places

A parking place accessory has a list of IDs for the vehicles that are present (either because they begin the simulation there or they have arrived during the course of the simulation). It also has a queue of travelers and their associated plans. This procedure handles each traveler in the traveler queue whose departure time has arrived.

If the traveler is waiting for a vehicle, check to see if the vehicle is present. If the vehicle is not there, increment the traveler's departure time and replace him/her in the queue. A vehicle whose ID is on the list will have been instantiated in the simulation only if it has arrived here from somewhere else. Otherwise, a new vehicle with this ID must be created using the type implied by the traveler's plan.

The traveler is added to the vehicle as a driver or passenger, depending on the traveler's plan. If the driver has not yet been added to the vehicle, pop the next traveler off the queue and continue. Otherwise, check to see how many passengers are anticipated. (This information is contained in the driver's plan, along with the IDs of the expected passengers.) If any passengers are missing, place the driver back in the queue so that the vehicle will try to leave again on the next timestep. If the driver and all passengers are present, try to place the vehicle on the grid in any lane, not in the boundary region, traveling at the speed limit.

The appropriate grid for the planned direction of travel is determined, and the grid is searched upstream for a distance of V_{Max} cells. If a vehicle is found in a lane, that lane and adjacent lanes are eliminated from consideration. All lanes are searched and if a lane is available, the vehicle is placed on the lane at the cell corresponding to the parking place location. If there is no room on the grid, the driver is returned to the traveler queue.

4.2.2.7 Movement Check / Intersections

This procedure handles vehicles that are leaving a link and passing through an intersection.

Upon arriving at an intersection, a vehicle's destination lane on the next link is determined. The current lane is chosen if it is allowed on the next link, otherwise a lane is picked at random from the set of allowed lanes. This set takes into consideration lane use and HOV restrictions.

Unsignalized intersections with stop/yield traffic controls require vehicles to consider oncoming traffic before they can move onto the next link. The vehicles use the gap between the oncoming vehicles and the intersection to determine whether the intersection can be entered. If the gap is acceptable, the vehicle traverses the intersection and arrives on the destination link during a single update step in the microsimulation.

Vehicles at signalized intersections have different behavior from those at unsignalized intersections. When a vehicle enters an intersection, it is placed in a queued buffer where it resides for a specified time before exiting to the destination link. The time that the vehicle spends in the queued buffer models the time necessary to traverse the intersection. Vehicles with permitted, but not protected, movements from the intersection traffic control must consider the oncoming traffic before entering the intersection.

In order to enter an intersection, the vehicle must satisfy the following:

- 1) Be the last vehicle on the link in the current lane going toward the intersection. Only one vehicle per lane is allowed to enter the intersection in a single timestep.
- 2) Have a current speed \geq the number of empty cells between the vehicle and the end of the link.
- 3) Satisfy the conditions of the traffic control at the intersection. The state of the traffic control indicates whether a vehicle must consider the oncoming traffic gaps.
- 4) Ensure that the gap between the vehicle and oncoming traffic is acceptable.
- 5) Ensure that the intersection buffer for the current lane is not full.
- 6) Ensure that the destination cell in the destination lane on the destination link is unoccupied.

A vehicle will attempt to enter an intersection if its current speed is \geq the number of empty cells between the vehicle and the end of the link. The destination lane on the next link is determined. If possible, the destination lane is the same as the current lane. If not possible, a random lane selection is used.

The state of the traffic control (TC) at the intersection is an important factor in the decision on whether the vehicle can enter the intersection. At a signalized intersection, the TC must indicate a permitted, protected, or caution movement for the current lane in order to enter the intersection. At an unsignalized intersection, stop and yield signs impose conditions on intersection entry. The TC state may require that the distance between the intersection and the on-coming traffic (interfering lane gap) meet certain criteria before the vehicle can enter the intersection. Table 29 shows the TC states and their corresponding actions.

Table 29: Traffic control states and corresponding actions.

TC State	Action	Conditions
S* - Protected	Proceed	None
S - Wait	Stop	None
S - Permitted	Evaluate	G_i on IL (Interfering Lanes)
S - Caution	Proceed	None
U** -None	Proceed	None
U - Stop	Wait	Stopped < 1 Timestep
	Evaluate	G_i on IL, Stopped \geq 1 Timestep
U - Yield	Evaluate	G_i on IL

* S = Signalized intersection

** U = Unsignalized intersection

The interfering lane gap (G_i) is the distance between the oncoming vehicle and the intersection. The oncoming vehicle must be on a link connected to the intersection, which limits the look-back distance for oncoming traffic to the length of a single link. The speed of the oncoming vehicle (V_{ov}) and the Gap Velocity Factor (GVF, specified by the configuration key CA_GAP_VELOCITY_FACTOR) are used to calculate the Desired Gap.

$$\text{Desired Gap } (G_d) = V_{ov} * \text{GVF}$$

On links where the desired gap is greater than the number of cells on the link, the number of cells on the link is used as the desired gap.

$$G_i \geq G_d, \text{ Interfering Gap Acceptable}$$

$$G_i < G_d, \text{ Interfering Gap Not Acceptable}$$

Note that for an oncoming vehicle with speed of 0, G_d will be 0, which allows movement through intersections in congested conditions where both G_d and $G_i = 0$.

If the interfering gap is not acceptable and the vehicle is at a stop or yield sign and the interfering lane is also controlled by a stop/yield sign, there will be a deadlock resolution in which the vehicle will proceed with probability determined by the value of the configuration key CA_IGNORE_GAP_PROBABILITY.

The vehicle can enter the intersection only when the interfering gaps are acceptable ($G_i \geq G_d$) (Figure 21).

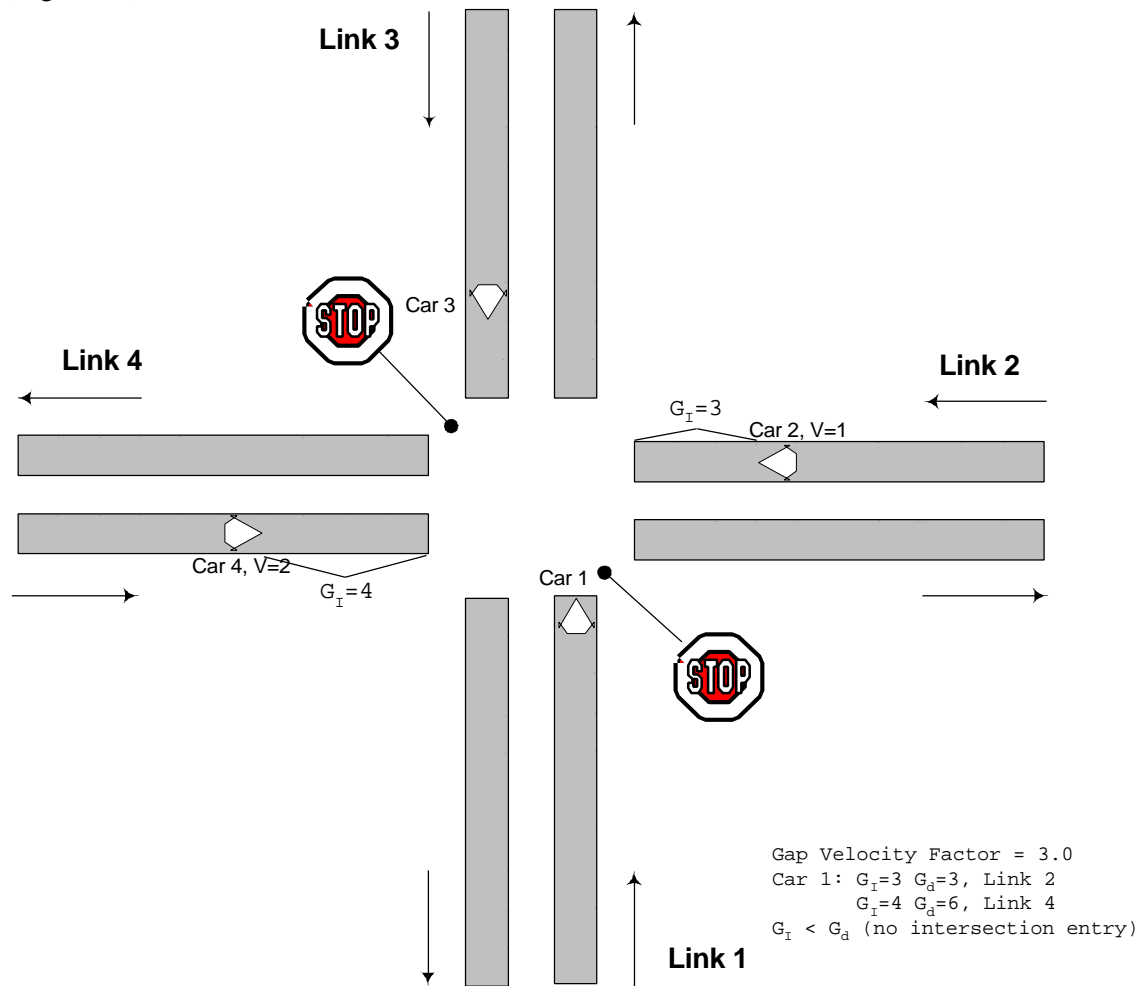


Figure 21: Intersection entry interfering lane gap.

If the traffic control for the intersection is signalized, the vehicle does not traverse the intersection in the current microsimulation timestep. Signalized intersections maintain internal queued buffers where vehicles are placed to traverse the intersection. Each intersection has one queued buffer for each incoming lane. If the conditions of the signalized traffic control have been satisfied, a vehicle must check whether the appropriate buffer has space to receive the vehicle. (The capacity of an intersection buffer is set by the configuration key `CA_INTERSECTION_CAPACITY`.) If so, the vehicle is removed from the incoming link and placed in the intersection buffer for a wait period (specified by the configuration key `CA_INTERSECTION_WAIT_TIME`). After the time period has expired, the vehicle will exit from the buffer to the first cell on the destination link if the cell is vacant. If not, the vehicle waits in the intersection buffer until the cell becomes vacant. The buffers have a fixed size, so that if the buffer is full, the vehicle cannot enter the intersection and must wait on the link.

At unsignalized intersections, the vehicles are capable of entering and exiting the intersection in a single timestep. Therefore, if the conditions of the unsignalized traffic control have been satisfied for intersection entry, a vacant cell on the destination link in the destination lane must be available

for the vehicle to enter the intersection. The vehicle's current speed is used to determine which cell to reserve on the destination link. If the primary destination cell is unavailable, the next cell closer to the intersection is tried. The process continues until an available cell is found, or until all of the cells between the intersection and the primary destination cell are tried. A marker is placed in the destination cell to reserve the cell.

If the vehicle succeeds in reserving a place in the queue or on the next link, an internal state variable will be set to indicate that it can proceed. This variable is used during the movement procedure to determine whether to remove a vehicle from a link or decrease its speed. Vehicles traversing unsignalized intersections are placed on their destination link during the clean-up procedure at the end of a timestep.

4.2.2.8 Off-Plan Vehicles

A vehicle that is not in an acceptable approach lane when it is ready to enter an intersection cannot follow its assigned plan; it is marked as an *off-plan vehicle*. Vehicles that have not moved for the number of timesteps defined by the configuration key `CA_MAX_WAITING_SECONDS` also become off-plan. The timestep when the vehicle will try to exit from the simulation is calculated using the off-plan exit time described in Section 3.4, and a new destination link is chosen from the links that are connected to the vehicle's current lane.

New destination links are randomly chosen for off-plan vehicles until the current timestep is equal to the calculated exit timestep. Then, the vehicles are removed from the simulation at the nearest parking place.

4.2.2.9 Abandon Plan

Vehicles that are trying to enter an intersection and have not moved for a specified period of time abandon their plans and, if possible, choose a different destination link.

`CA_MAX_WAITING_SECONDS` defines the time period (Section 2.8). These vehicles are marked as off-plan and are removed at the nearest parking place. Allowing vehicles to become off-plan after a specified waiting period is necessary to prevent traffic gridlock in congestion.

4.2.2.10 Movement

The movement rule is "Accelerate when you can; slow down if you must; sometimes slow down for no reason." The rule is executed to update the speed and position of each vehicle on the roadway.

The distance between a vehicle and the next car ahead is called the *gap*. Each vehicle will try to accelerate if the gap is greater than the desired speed. The desired speed is limited to the speed limit posted on each link. If the gap is smaller than the current speed, the vehicle will slow down until its current speed is equal to the gap, thus imposing the no-collision condition. Each vehicle also has a random probability of slowing down. This is called the deceleration probability (P_d). Use of the deceleration probability is essential to produce realistic traffic dynamics, such as jam waves, from the individual vehicle interactions.

To compute a vehicle's speed (V_{t+1}) and the next position on a link, first compute the speed based on the gap and the vehicle's speed in the current timestep (V_t).

- 1) Compute Gap

- 2) if $(V_t < \text{Gap AND } V_t < V_{Max})$

$$V_{t+1} = V_{t+1}$$

Each moving vehicle ($\text{Speed} > 0$) has a random probability of decelerating in each timestep. Compute the probability and slow down if the computed probability is less than the deceleration probability.

- 3) if $(V_{t+1} > 0)$ and $(N_{Rand} < P_D)$

$$V_{t+1} = V_{t+1} - 1$$

Next, move the vehicle to its new grid position based on the new speed.

- 4) New Cell = Current Cell + V_{t+1}

4.2.2.11 Entering Parking Places

Vehicles are removed from the roadway at destination parking places by checking all of the cells in all lanes downstream from a parking place for a distance of $V_{GlobalMax}$ cells. If a vehicle is found on the last step of the current leg of its plan and with this parking place as its destination, the vehicle is removed from the roadway. Its ID is placed onto the list of vehicles present at that parking place.

4.2.3 Preparing for a Timestep

4.2.3.1 Update Signals

Signals are updated at each timestep according to the timing table provided for each signal. This version of TRANSIMS implements pre-timed signals only. Signalized traffic controls are initialized at the beginning of the simulation to the first interval of the first phase of the signal cycle when the signal offset is 0.0. When the offset is non-zero, the signal is initialized to the phase and interval that corresponds to simulation time 0 in the offset cycle.

4.2.3.2 Prepare Nodes

Find vehicles in each intersection that are ready to be ejected during this timestep. Vehicles exit from the intersection queued buffers when their residence time in the buffer is greater than the intersection residence time specified by the configuration key `CA_INTERSECTION_WAIT_TIME`. Vehicles exit from the queued buffer onto the first cell in the destination lane on the destination link. Exiting vehicles reserve their destination cell before vehicles on links calculate movement, which gives the vehicles exiting from intersection buffers precedence over vehicles on the links. Vehicles are transferred from the buffers to their reserved destination cells during the clean-up phase after movement changes for all of the vehicles are executed. The speed of the vehicle does not change during intersection entry/exit at a signalized intersection. Vehicles are placed in the first cell on the destination link with the same velocity that they entered the intersection buffer.

4.2.4 Prepare Edges

`PrepareEdges` does nothing in the current version of TRANSIMS, unless it is compiled with `CA_CACHE_LANE_TRAFFICCONTROLS` defined. In that case, it caches the signal values for each lane on the grid.

4.2.5 Cleaning up After a Timestep

4.2.5.1 Migrate Vehicles

Any vehicle that has passed from the region of a link controlled by a CPU into the region controlled by its neighbor must be encoded in a message and sent to that neighbor, as described in Section 2.6. This is done on a link by link basis.

4.2.5.2 Migrate Travelers

Some travelers who are not in vehicles may have been placed in the *Migrating Travelers* list during the timestep. This procedure encodes those travelers into messages and passes them on to the desired CPUs, clearing the list out as it goes.

4.2.5.3 Clean Up Nodes

This procedure causes each intersection to eject the first vehicle in each of its buffers into previously reserved locations on the destination link.

4.2.5.4 Clean up Edges

This procedure clears all temporary vehicle markers from the grids. Also, if the *clean up action* state variable for a vehicle is *eject*, it places the vehicle in the intersection buffer (if buffered, otherwise place it directly onto the next edge). If the clean up action is *migrate*, it deletes the vehicle (which has already been sent to its destination CPU in the migration step).

4.2.6 Supporting Parallel Computation

The Traffic Microsimulator runs on multiple CPUs to maximize the computational speed. Updating of vehicle positions can then be done in parallel on the individual CPUs. This method is faster than a single, sequential update algorithm on transportation networks with a large number of vehicles.

4.2.6.1 Transportation Network Partition

The transportation network is partitioned among the CPUs, with each CPU receiving a set of nodes and links (Figure 22). The network can be partitioned among the CPUs using either an orthogonal bisection algorithm or the METIS graph partitioning library. METIS is a public domain package. Which algorithm is used is determined at run time by a combination of the configuration keys `PAR_PARTITION_FILE`, `PAR_USE_METIS_PARTITION`, and `PAR_USE_OB_PARTITION`.

Both algorithms use a cost function for each node. METIS also uses a cost function for each link. These costs can be based on the number of cells on the links attached to the node if no other information is available. As the simulation runs, it collects information on the amount of CPU time devoted to processing each link and node. This information can be saved in a *Run Time Measurements* file, which can be used to assign costs to the links and nodes in subsequent partitioning calculations. The configuration keys that control this process are PAR_RTM_INPUT_FILE, PAR_RTM_PENALTY_FACTOR. The output file name is currently hard-coded to be RunTimeMonitor. It is placed in the directory named by OUT_DIRECTORY.

The partitioning can be saved for later use by *DistributePlans* or a subsequent simulation run. This is controlled by the configuration key PAR_SAVE_PARTITION and PAR_PARTITION_FILE. If neither METIS nor OB partitioning has been requested, the simulation will look for this partition file.

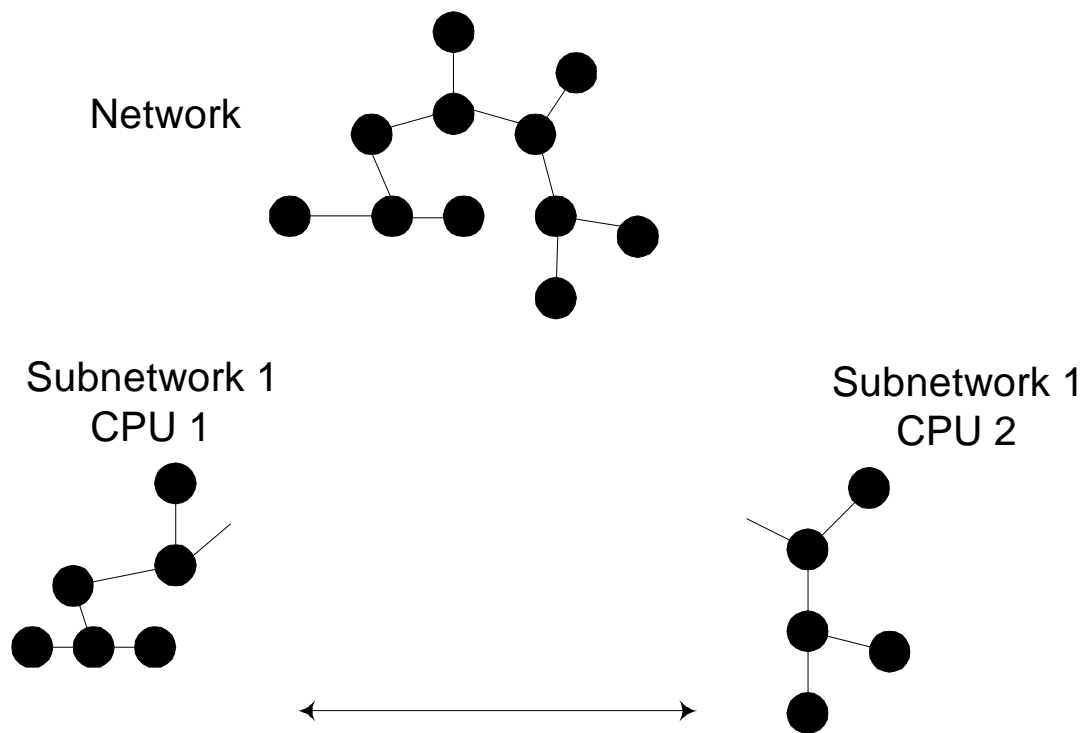


Figure 22: Transportation network partition.

4.2.6.2 Distributed Links and Boundary Information Flow

Links that connect nodes that reside on different CPUs are split in the middle (Figure 22). These links are distributed links. Each CPU is responsible for one-half of the link. Each distributed link is assigned the number of active grid cells belonging to the given CPU. This is necessary to accurately divide links with an odd number of cells. The area in the middle of the distributed links is called a boundary area. The width of the boundary area is currently $V_{GlobalMax}$ (5) cells. Links that are shorter than PAR_MIN_CELLS_TO_SPLIT cells will not be split. The maximum distance (forward or backward on a link) that can be used for gap calculations is limited to the boundary width on distributed links.

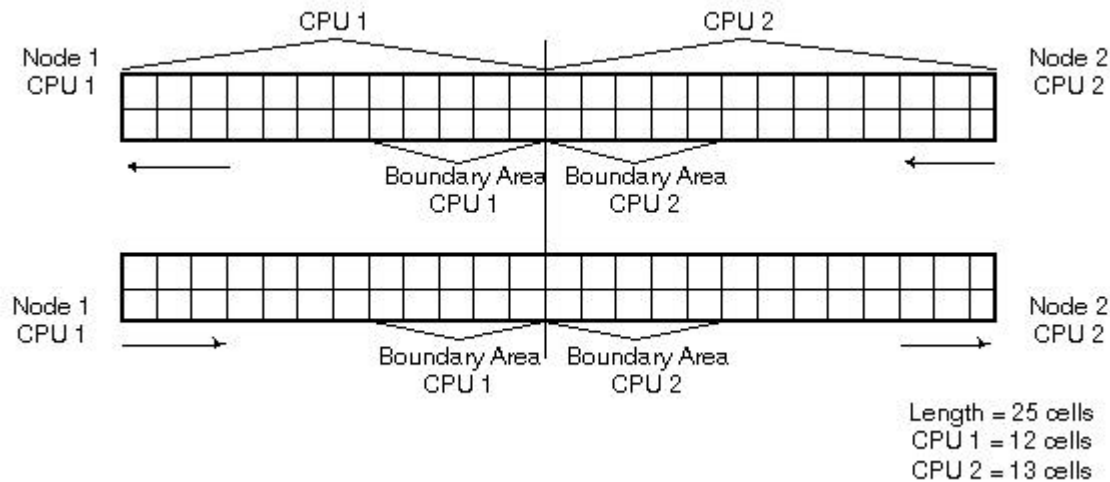


Figure 23: Distributed (split) link.

Vehicles are transferred between the CPUs as they traverse these split links. Each split link introduces a message-passing delay during the update sequence because messages must be passed between the CPUs for vehicles that are crossing the split links. Two types of messages must be exchanged between CPUs with distributed links.

- 1) Vehicle Migration Messages – Messages for vehicles transferred to the other part of the link on a different CPU.
- 2) Boundary Exchange Messages – Messages containing information about vehicle positions in the boundary area of a link.

Vehicle migration messages occur for all vehicles that have completed traversal of a CPU's active cells. All information about the vehicle, its occupants, and their plans is put into a message and sent to the CPU that owns the other half of the distributed link, after which the vehicle is removed from the originating CPU. Upon receipt of the message, the other CPU creates a vehicle and travelers using the information in the message and places them at the appropriate position on its half of the distributed link.

Exchange of boundary information between CPUs is called a boundary exchange. Boundary exchange messages are necessary to correctly calculate position changes (movement and lane changes) for vehicles in a CPU's boundary area. Information about vehicles in the next $V_{GlobalMax}$ cells (or preceding $V_{GlobalMax}$ cells, depending on the direction of traffic flow) is necessary to execute the appropriate gap calculations for lane changes and movement. Each CPU maintains a list of its distributed links and of the CPU owners of the other half of the links. Boundary exchanges must be done before lane changes and again before vehicle movement. The exchanges are initiated by each CPU at the appropriate time. Each CPU waits until it receives all of the boundary exchange messages from neighboring CPUs.

4.2.6.3 Parallel Computation Sequence and Synchronization Points

The Traffic Microsimulator is a distributed object simulation using a master/slave(s) paradigm. The master process starts the slave processes, handles the initialization sequence, and then serves as a synchronization point for the slave processes. The slave processes do all of the work in the

simulation. After initialization, each slave process completes successive update cycles until the end of the specified simulation run. The slave processes synchronize with the master process at the beginning of each timestep or at the beginning of a sequence of timesteps, depending on the value of the configuration key CA_SEQUENCE_LENGTH (see Figure 24).

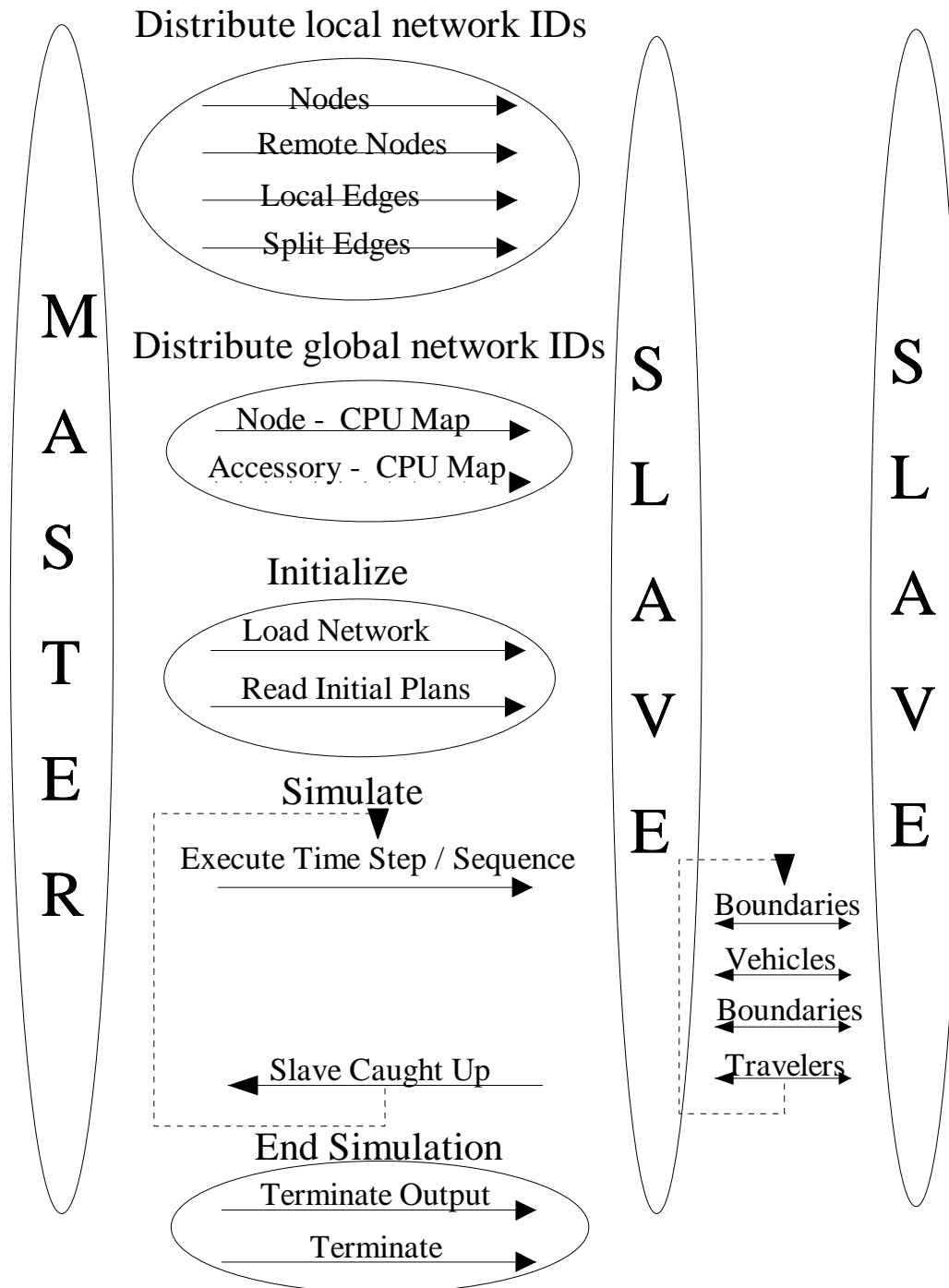


Figure 24: Communication in the parallel simulation.

4.2.6.3.1 Initialization Sequence

The master process starts first, reads network information from the database, constructs a copy of the transportation network, and constructs or reads a partition. The master is then ready to create and initialize the slave processes using the following sequence:

- 1) Start slave processes.
- 2) Send each slave lists of the IDs of its local nodes and links and of those connected to it by distributed links.
- 3) Send each slave a mapping from node IDs to CPU IDs, and optionally (depending on the setting of the configuration key `CA_BROADCAST_ACC_CPN_MAP`), a mapping from accessory IDs to CPU IDs.
- 4) Tell each slave to construct its transportation subnetwork from information in the database.
- 5) Tell slaves to read in the initial plans, to queue initial vehicles on parking places, and to do initial placement of vehicles on the links at the given simulation start time.

4.2.6.3.2 Simulation

After the initialization sequence is complete, the master starts the simulation by telling the slaves to execute the first timestep. The master process waits until all of the slaves complete execution of a fixed number of timesteps then sends a message to the slaves to execute the next sequence of timesteps.

4.2.6.3.3 Termination

The master sends messages to the slaves to shut down the parallel I/O system and then to exit when the requested number of timesteps has been executed.

4.2.6.4 Overlapping Computation and Communication

It is generally a good idea to overlap communication and computation in a parallel code. This allows a CPU to continue doing useful work while waiting for responses from other CPUs.

TRANSIMS accomplishes this by noting which links are under a single CPU's control and which are shared. After sending boundary information, each CPU can update all of its non-shared links before it must make use of boundary information received from other CPUs. Figure 25 shows the sequences of computation and communication in such a modified timestep execution. If the configuration key `CA_LATE_BOUNDARY_RECEPTION` is set, the simulation will arrange computations in this manner.



Figure 25: Order of execution of processes in a modified timestep.

4.2.6.5 Output Collection

All output information from the Traffic Microsimulator is generated in parallel by the slave(s). Each slave sends a message to the master indicating what sort of information it would like to write and how many bytes the information will require on disk. The master collates the requests from all the slaves and responds to each, indicating an offset into a file for writing the information. Each slave then writes its information to disk at the indicated location.

4.3 Usage

4.3.1 Configuration Parameters

The configuration parameters (Table 30) control how drivers and vehicles behave in traffic. Variations in behavior among drivers is accomplished by allowing certain behaviors to vary randomly within limits.

Table 30: Configuration Parameters.

Configuration Parameter	Description
CA_DECELERATION_PROBABILITY	Variation in the traffic is enhanced by having each driver randomly decide whether to decelerate for no apparent reason at each timestep. The probability of decelerating is a value in the range 0.0 to 1.0 [default = 0.2].
CA_LANE_CHANGE_PROBABILITY	Variation in the traffic is reduced by not allowing every driver who would change lanes based on vehicle speed and gaps in the traffic to do so at each timestep. This is done to prevent <i>lane hopping</i> . The probability that a driver will change lanes when speed and gaps permit is a value in the range of 0.0 to 1.0 [default = 0.99].
CA_PLAN_FOLLOWING_CELLS	Plan Following Cells specifies a count of the number of cells preceding the intersection within which a vehicle will make lane changes to get in an appropriate lane to transition to the next link in its plan. Beyond this distance, lane changing decisions are based only on vehicle speed and gaps in the traffic. Within this distance, the lane required by the vehicle's plan is also taken into account. As the vehicle nears the intersection, the bias to be in the lane required to stay on plan is increased. Valid values are positive or zero [default = 70 cells].
CA_LOOK_AHEAD_CELLS	The preferred lane for a vehicle to be in as it approaches an intersection depends on the connectivity from the current link to the next link in the plan. In some situations, it is advantageous for the driver to look beyond the next link to subsequent links in the plan when deciding the preferred lane. Look Ahead Cells controls how far ahead the driver will look. A value of 0 indicates that the driver will not look beyond the next link. A positive value indicates that the driver will look at least one additional step beyond the next step in the plan. The number of additional links that will be considered is determined by the lengths of the subsequent links, with link lengths being summed until the accumulated distance is greater than or equal to Look Ahead Cells. Valid values are positive or zero [default = 35 cells].
CA_INTERSECTION_WAIT_TIME	Intersection Wait Time specifies the number of seconds that a vehicle requires to pass through a signalized intersection. A vehicle resides in an intersection queued buffer for this amount of time and is then placed on the next link if the first cell on that link is unoccupied. It will remain in the intersection for a longer time if entry to the next link is blocked by another vehicle. Valid values are positive [default = 1 second].
CA_OFF_PLAN_EXIT_TIME	Off Plan Exit Time specifies the number of seconds a vehicle is allowed to deviate from its plan before being removed from the simulation. This prevents off-plan vehicles from wandering on the transportation network. Valid values are positive [default = 1 second].

Configuration Parameter	Description
CA_IGNORE_GAP_PROBABILITY	Drivers at unsignalized intersections wait for a suitable gap in cross traffic before proceeding through the intersection. The deadlock that would occur when vehicles are waiting for each other at multiway stop/yield signs is prevented by allowing each driver to ignore the gap constraint with some probability. The probability that the drivers at multiway stop/yield signs will ignore the constraint is a value in the range of 0.0 to 1.0 [default = 0.66].
CA_GAP_VELOCITY_FACTOR	At unsignalized intersections and during protected movements at signalized intersections, drivers wait for a suitable gap in cross traffic before proceeding through the intersection. The number of empty cells in a suitable gap is based on the speed of the cross traffic and the gap velocity factor. The suitable gap is calculated for each lane of the cross traffic. $\text{Gap} = \text{Speed of Oncoming Vehicle} * \text{Gap Velocity Factor}$ The gap velocity factor must be greater than 0.0. The default value is 3.0. Note that vehicles with a speed of 0 result in a suitable gap size of 0, which improves traffic flow in congested conditions.
CA_MAX_WAITING_SECONDS	Max Waiting Seconds determines the number of seconds that a vehicle will try to enter an intersection. If the vehicle has not moved from the link into or through the intersection in Max Waiting Seconds, the vehicle will abandon its plan and try an alternative movement through the intersection, if one exists. Max Waiting Seconds must be greater than 0 and should be greater than the longest red phase of the traffic controls in the simulation. The default value is 600 seconds.
CA_INTERSECTION_CAPACITY	Intersection Capacity determines the number of vehicles that can be held by each intersection's buffers. This is interpreted as the effective number of cells available in the buffer; that is, a vehicle that is 2 cells long will not fit in an intersection if its capacity is 1.
CA_MAXIMUM_SPEED CA_MAXIMUM_ACCELERATION	Maximum speed (in cells / timestep) and acceleration (in cells/timestep/timestep) are applied to vehicles of type auto in the movement phase of traffic dynamics.
CA_BUS_LENGTH CA_BUS_CAPACITY CA_BUS_MAXIMUM_SPEED CA_BUS_MAXIMUM_ACCELERATION	Bus Length specifies the length (in cells) of every transit-type vehicle. Bus Capacity specifies the maximum number of occupants (including the driver) allowed in a transit-type vehicle. Bus Maximum Speed and Acceleration are the same as Maximum Speed and Acceleration, except these are applied to transit-type vehicles.
CA_TRANSIT_INITIAL_WAIT	Transit Initial Wait specifies the number of timesteps a transit vehicle must be present at a transit stop before any passengers get on or off.
CA_ENTER_TRANSIT_DELAY CA_EXIT_TRANSIT_DELAY	These specify the mean number of timesteps it takes for a single traveler to enter or exit a transit vehicle.
PLAN_FILE	The plan file specifies the name of the file where plans reside or a string to which <i>.tim.idx</i> and <i>.trv.idx</i> can be appended to find the time-sorted and traveler-id-sorted indexes into a plan file(s). The plans should include all travelers—for example, plans created by the Route Planner, transit driver plans, freight plans, etc. The name should be given as an absolute path name, since the slave executables are not always run from the current working directory.
VEHICLE_FILE	The vehicle file specifies the name where vehicles reside, or a string to which <i>.veh.idx</i> can be appended to find the vehicle-id-sorted index into a vehicle file(s). The vehicle file must include all vehicles to be used in the simulation.

Configuration Parameter	Description
CA_USE_PARTITIONED_ROUTE_FILES	It is more efficient for slaves to read only those plans that start in the part of the network for which they are responsible. If the partitioning to be used by the simulation is available (from a prior run of the simulation, for example), the <i>DistributePlans</i> utility will create a separate pair of indexes for each slave into one common plan file. If Use Partitioned Route Files is set, the slaves will look for these slave-specific indexes. If they do not exist, the simulation will fall back to using a single global pair of indexes.
CA_NO_TRANSIT	If this flag is set, travelers whose plans originate or end at a transit stop are removed from the simulation. None of their remaining legs are used. (The transit drivers plans do not fall into this category, thus transit vehicles can still be present in the simulation, but no passengers will use them.)
CA_RANDOM_SEED1 CA_RANDOM_SEED2 CA_RANDOM_SEED3	These three values are combined to initialize the random number generator. Note that the actual sequence of random numbers generated on a slave also depends on the number of slaves and the partitioning in general.
CA_SIM_START_HOUR CA_SIM_START_MINUTE CA_SIM_START_SECOND	These values are combined to give the time of day at which the simulation starts. Plans whose estimated arrival time is before the start time will not be executed.
CA_SIM_STEPS	The simulation executes Sim Steps timesteps before exiting.
CA_SEQUENCE_LENGTH	The slaves are implicitly synchronized among themselves by the actions of passing boundaries and migrating vehicles. They are also explicitly synchronized by the master every Sequence Length timesteps. It may be more efficient to allow the implicit synchronization to control the simulation.
CA_SLAVE_PRINT_MASK CA_MASTER_PRINT_MASK	These variables control which logging messages to ignore. They are set within the code based on the values of the LOG_ configuration keys and should not be set directly.
CA_SLAVE_MESSAGE_LEVEL CA_MASTER_MESSAGE_LEVEL	Only warning messages whose severity is at least as high as Message Level will be written to the master or slave's log file.
CA_USE_NETWORK_CACHE	If set, use a cached binary representation of the network. This representation would have been created by a prior run of the simulation.
CA_OUTPUT_BUFFER_SIZE CA_OUTPUT_BUFFER_COUNT CA_OUTPUT_WRITE_INTERVAL	The parallel I/O system buffers output from each slave until the master provides a file offset to which it can be written and Output Write Interval timesteps have occurred. The Output Buffer Size determines the size of these buffers in bytes. The Output Buffer Count determines the maximum number of buffers that can be opened for each output stream. If the total amount of data buffered exceeds the product of the size and count, the simulation will exit.
CA_USE_ROMIO_FOR_OUTPUT	If Use Romio For Output is set, and the executable was compiled with the USE_ROMIO and USE_MPI flags defined, the parallel output system will use ROMIO files instead of UNIX files
CA_PRESORT_EVENT_FILES	Event output is not necessarily produced in globally time-ordered fashion by the different slaves. If this flag is set, the event file output should be correctly sorted by simulation time.
CA_LATE_BOUNDARY_RECEPTION	If Late Boundary Reception is set, the simulation will try to overlap computation and communication as described above in Section 4.2.6.4.

Configuration Parameter	Description
CA_RTM_SAMPLING_INTERVAL CA_RTM_OUTPUT_FILE	The partitioning algorithms try to find the partition that spreads the computation associated with nodes and links evenly while simultaneously trying to minimize the communication costs associated with split links. The costs for each node and link can be estimated using run time costs from prior runs. These costs are sampled at the interval defined by RTM Sampling Interval and written out to the file named by RTM Output File. They are currently read in from a file whose name is hard-coded to be RunTimeMeasurements, which is expected to be found in the directory named by OUTPUT_DIRECTORY.
CA_BROADCAST_ACC_CPN_MAP CA_BROADCAST_TRAVELERS	As described above in Section 4.2.5.2, if Broadcast Travelers is set, migrating travelers are broadcast to every CPU. Since only one CPU will eventually make use of the traveler, this is inefficient. If Broadcast Acc CPN Map is set, each CPU knows which CPU is associated with every accessory, so traveler migration messages can be targeted to only the single CPU that needs them. If the CPN Map is not broadcast, travelers must be broadcast.
PAR_SLAVES	This key sets the number of slave processes to spawn. It must be smaller than the number of host CPUs available (to allow one process for the master).
PAR_HOST_COUNT	The number of distinct machines that make up the parallel machine environment.
PAR_HOST_I PAR_HOST_CPUS_I PAR_HOST_SPEED_i	These variables describe the parallel machine environment to the simulation. There should be one set of these three variables, with <i>i</i> replaced by an integer from 0 to the value of PAR_HOST_COUNT - 1, for each host. Host should be a string containing the name of the machine. Host CPUs should give the number of CPUs available for use on the machine. Host Speed should give the relative speeds of the different machines in arbitrary units. The sum of all the values of Host CPUs must be at least one larger than the number of slaves requested.
PAR_BROADCAST_CONFIG	Broadcast parallel configuration information to all hosts. This must be set to use MPI.
PAR_MPI_SPIN	Activates spinning while waiting for messages.
PAR_COMMUNICATION	One of "MPI" or "PVM." This is evaluated only by the script Msim.pl, which can be used to run the simulation.

4.3.1.1 Parallel Configuration and Communication Setup

The Traffic Microsimulator can use either of two parallel execution environments: Parallel Virtual Machine (PVM) or Message Passing Interface (MPI). Both are widely available with no license fee. Both provide a communication mechanism between processes. These processes can be on the same CPU/host or on different CPUs/hosts. Each microsimulation master/slave process becomes a task. Messages between master and slave(s) and among slaves utilize PVM or MPI as the underlying message handler.

The simulation can be run on a single processor, using a single slave, but it still requires either PVM or MPI for communication between the master and slave process.

The script *Msim.pl* will execute the simulation under either PVM or MPI. This script requires that the environment variable TRANSIMS_HOME be set to the name of the top-level TRANSIMS directory, which includes the subdirectories *scripts*, *source*, *pvm*, etc. It takes the path name of a

configuration file and the name of a log file as its only required arguments. In addition, the configuration file key PAR_COMMUNICATION or an environment variable with the same name should be given the value “MPI” or “PVM,” depending on which environment is desired.

Msim.pl makes a reasonable guess at values for the required information for running under either PVM or MPI. This information consists of the root directory for the executables and libraries, a string designating the architecture to be used, and for MPI, a string designating the name of the device to be used. See the documentation supplied with PVM and MPI for further description of allowed architecture and device strings.

The script’s guess for each of these can be overridden by providing values for the following keys, either in the configuration file or the environment: PAR_PVM_ARCH, PAR_PVM_ROOT, PAR_MPI_ARCH, PAR_MPI_ROOT, PAR_MPI_DEVICE. If values are provided in more than one place, the precedence is configuration file, environment, default value.

The *Makefile* system creates different versions of the simulation, placed in communication environment, architecture, and device specific directories. *MSim.pl* will choose an appropriate version of the executable. The choice may be overridden by specifying a value for CA_BIN in the configuration file or the environment.

Both PVM and MPI require that the user be able to gain access to the machine on which processes are to be run without typing a password. This is most easily accomplished through the use of a file named *.rhosts* in the user’s home directory. This file should list the name of the machine to be used and the user’s login name on that machine. The file’s permissions should be set to deny access to others. You should be able to remote shell (*rsh*) to the desired machine without being prompted for a password. Because of potential security risks, some networks automatically remove *.rhosts* files on a daily basis, so it may be necessary to recreate it. Contact your system administrator for details on the most appropriate setup for your system.

The simulation may also be run under MPI or PVM directly from the command line. For more information on starting processes in these environments, consult the PVM or MPI documentation. For PVM, start a *pvm* daemon, then just type the full pathname of the executable and the full pathname of a configuration file. For MPI, use *mpirun* and supply the number of processors, which should be one more than the number of slaves specified in the configuration file. See the examples in the tutorial below.

4.4 Tutorial

All these examples assume that the environment variable TRANSIMS_HOME is set to the directory containing the TRANSIMS distribution.

Here is an example using the *Msim.pl* script to run the simulation:

```
% $TRANSIMS_HOME/scripts/Msim.pl $TRANSIMS_HOME/config/Calnet_default logfile
```

Here is an example using PVM directly under *csh*, assuming the PVM architecture is SUN4SOL2. First add the following lines to your *.cshrc* file, so that all the processes started by PVM get the same environment:

```
setenv PVM_ROOT $TRANSIMS_HOME/pvm/pvm3
set path = ( $PVM_ROOT/bin $PVM_ROOT/lib $path )
```

Then execute:

```
% pvm
pvm> quit
Console: exit handler called
pvmd still running
% $TRANSIMS_HOME/bin/ARCH.PVM.SUN4SOL2/CA $TRANSIMS_HOME/config/Calnet_default
```

Finally, here is an example using MPI directly, with 4 slaves, on the MPI architecture “solaris,” with the MPI device type `ch_p4`. First add the following lines to your `.cshrc` file, so that all the processes started by MPI get the same environment:

```
setenv MPI_ROOT $TRANSIMS_HOME/mpich
set path = ( $MPI_ROOT/bin )
```

Then execute the simulation:

```
% mpirun -np 5 $TRANSIMS_HOME/bin/ARCH.MPI.solaris.ch_p4/CA $TRANSIMS_HOME/config/Calnet_default
```

4.5 Troubleshooting

4.5.1 Miscellaneous debugging strategies

- 1) Turn on all logging flags (those that start with `LOG_`).
- 2) Set the Message Level parameters to 0.

4.5.2 ERROR in [PAR]: maximum number of saved buffers (64*1024 bytes) for PIOID 6 reached

Understanding this error requires some knowledge of how the Parallel IO system works,

- 1) The slave fills up a buffer with information.
- 2) When the buffer is full, the slave sends a message to the master requesting an offset into a file.
- 3) The master reserves room in the file for the buffer and replies with offset.
- 4) The slave receives offset (as an event) and stores it.
- 5) When the slave calls `WriteSavedBuffers` (called from `ParallelWrite` or `TerminateServices`), all saved buffers with saved offsets are written into the output file.

The slave calls `ParallelWrite` once every `CA_OUTPUT_WRITE_INTERVAL` timesteps. If the buffers overflow during a single timestep, the microsimulation crashes.

For example, a summary output processor loops over links adding data to its buffers for each link. If there is more data than (approximately) `CA_OUTPUT_BUFFER_SIZE * CA_OUTPUT_BUFFER_COUNT`, the code dies during the loop. This cannot be remedied by adding calls to `ParallelWrite` during the loop because the events that pass offsets back to the slave are not interrupts—they are only processed at specific times during the simulation. Hence, no offsets are saved until after the loop finishes and `ParallelWrite` will not be able to write out any saved buffers.

If you get this error in the first timestep, just keep increasing the number or size of buffers until it goes away. Buffers are only allocated as needed, and the same size and maximum number of buffers are used for every output stream, so you might want to leave the buffer size small and increase the maximum number to avoid memory problems.

To find out what output stream is causing the error, turn on LOG_IO and look for a message like the following in the slave's log file:

TParallelIO: opened UNIX file '/home/Gershwinoutput1/kpb4jh/output.test.snap.sig' for PIOID 6

If you get this error after several timesteps, your buffers are probably large enough to hold data from a single timestep, but they are not being flushed often enough. Reduce the CA_OUTPUT_WRITE_INTERVAL in your configuration file.

4.5.3 ERROR in [CA]: PIOID_MAX exceeded

This error occurs when the number of output files exceeds a hard-coded limit in the microsimulation. It can be fixed by turning off some of the output or recompiling with a higher value for PIOID_MAX.

4.5.4 ERROR in [PAR]: SIGNAL 11 encountered

This error can be caused by missing names for OUT_SNAPSHOT_NODES, or any other required file name in the configuration file.

4.5.5 ERROR in [PARENV]: Cannot open configuration file '/home/transims/release/output/Calnet/Configuration'!

The *Msim.pl* script adds a few configuration keys to those in the configuration file specified on the command line. It places both old and new keys in a new configuration file called *\$OUT_DIRECTORY/Configuration*. If this file already exists and the user does not have write permission on the file, the microsimulation will fail. There may be similar problems with other files in the OUT_DIRECTORY that are not writable by the user. This problem can be fixed by removing the files in OUT_DIRECTORY, by changing their permissions, or by overriding the value of OUT_DIRECTORY.

5. EMISSIONS ESTIMATOR MODULE

5.1 Overview

The purpose of the TRANSIMS Emissions Estimator module is to translate traveler behavior into emissions of nitrogen oxides, hydrocarbons, and carbon monoxide, as well as energy consumption, and carbon dioxide emissions. The calculated emissions can then be used with an airshed model to calculate pollutant concentrations for a metropolitan area

This section describes the Emissions Estimator module of TRANSIMS. The current version of TRANSIMS treats tailpipe emissions from light-duty vehicles; evaporative emissions and emissions from heavy-duty vehicles will be calculated in future releases.

The Light-Duty Vehicle Tailpipe submodule treats tailpipe emissions from cars, small trucks, and sport-utility vehicles. Important aspects include:

- 1) malfunctioning vehicles
- 2) emissions from cold starts
- 3) emissions from warm starts in which the engine is still warm but the catalyst is cold
- 4) emissions from off-cycle conditions which render the pollution controls inefficient
- 5) normal driving

With regard to off-cycle conditions, very high emissions occur at high power demands. The phrase off-cycle refers to conditions outside those that occur in the federal test procedure [1]. Emissions in this context are very sensitive to the precise acceleration that occurs at a specific speed.

The information flow of the Emissions Estimator module is summarized in Figure 26. The Emissions Estimator requires information on the fleet composition developed from the Population Synthesizer module, information on Inspection and Maintenance (I & M) test results, and traffic patterns. The traffic patterns are produced by the microsimulation.

The output of the system is emissions on the 30-meter segments for each 15-minute period simulated. Fuel economy and CO₂ emissions are also estimated. The emission inventory is designed to be used with EPA's MODELS-3 [2] to produce three-dimensional hourly gridded emissions over the metropolitan area

5.2 Light-Duty Vehicle Tailpipe Submodule

The Light-Duty Vehicle (LDV) Tailpipe submodule treats emissions from off-cycle driving, malfunctioning vehicles, normal driving, idling, and vehicles with cold engines and /or catalysts. The LDV Tailpipe submodule information flow is illustrated in Figure 26.

There are three major sets of information that must be developed:

- 1) what is the fleet composition?
- 2) what is the fleet status?
- 3) what is the fleet doing?

Once these questions are answered, the LDV Tailpipe submodule can produce the emissions.

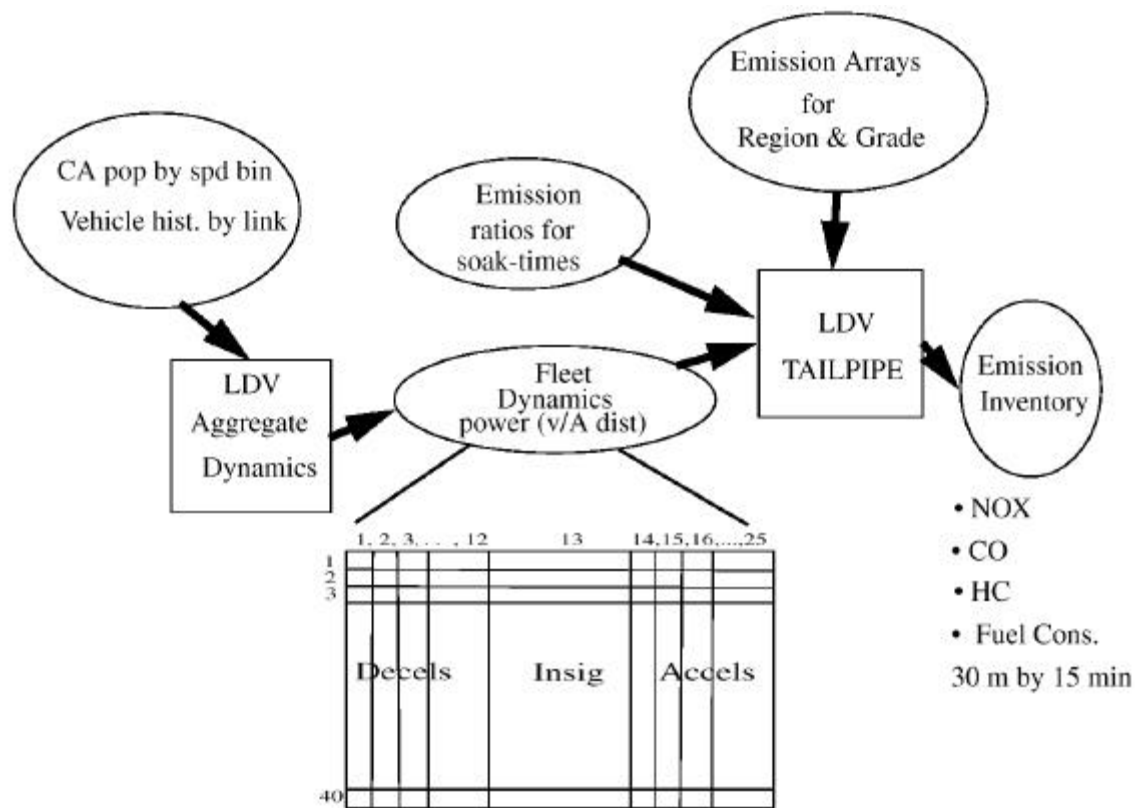


Figure 26: A flowchart of the process to produce a gridded, light-duty vehicle emission inventory.

5.2.1 Fleet Composition

Fleet composition is developed from vehicle registration data, inspection and maintenance testing, or data developed for EPA's mobile model runs. Barth and his colleagues [3] have developed techniques to take registration data and produce vehicle populations in each of 23 categories. The categories include factors such as low or high engine-to-weight ratio, car or truck, mileage above or below 50,000, type of catalyst (2-way or 3-way), carbureted or fuel-injected, and high emitting or normal emitting. In areas where there is a sophisticated inspection and maintenance program that tests vehicles on a dynamometer, improved estimates of the proportion of high-emitting vehicles can be made.

In the current version of the model, the vehicle distribution is used to compute the relationship between emissions and speeds and accelerations for a composite vehicle that is representative of the fleet in Southern California. The file *array.out* embodies these relationships and is used in the LDV tailpipe submodule.

5.2.2 Fleet Status

Fleet status is developed from the pattern of usage of the vehicles traversing a given link. The Traffic Microsimulator keeps track of when and where the vehicles have been operating. Cold engines burn fuel-rich until the engine has burned enough fuel to bring the engine temperature up to

normal. Similarly, the catalyst efficiency is reduced until enough fuel has been burned to bring the catalyst up to normal operating ranges. Within a given vehicle category, the principal determinant of fuel consumption is power demand.

We represent the distribution of vehicles in various stages of warm-up by grouping the vehicles entering a link into groups based on their integrated product of speed and acceleration since the last start of the engine and on the soak-time between the current operation and end of the last trip. In this version of the code, there is only a single soak-time and it was based on a one-hour time between starts. For NO_x , shorter soak-times have a larger effect, but the difference between cold and warm emissions is much less. There are seven groupings based on velocity-acceleration product, and there is an additional grouping for engines that have been fully warmed-up for a total of eight groups. The integrated velocity-acceleration product is in units of cells-squared per second squared; a cell is 7.5 meters.

For each group, we assign a multiplier for each parameter—hydrocarbons, carbon-monoxide, nitrogen-oxides, and fuel consumption. The multiplier represents the ratio of emissions for vehicles beginning a link in the group to the emissions of a vehicle with the same driving pattern with fully-warmed up engine and catalyst. The file *wcemratios* provides these ratios. The first column represents hydrocarbon emissions, the second carbon-monoxide, the third nitrogen-oxides, and the fourth fuel consumption. The first row gives the soak time of 60 minutes and the lowest integrated velocity-acceleration product. The integrated velocity-acceleration product increases with row until the eighth row, which is all ones because it is the ratio of emissions from warmed-up vehicles to emissions of warmed-up vehicles.

The groupings and values were obtained by taking several actual trajectories where vehicles accelerated from a near stop (less than 5 mph) at a signal and achieved typical speeds on an arterial. The original trajectories were approximately 30 seconds long and covered about one-quarter mile. The ends of these trajectories were replaced with a short-deceleration to the initial speeds, and then the trajectories were repeated. In this way, several trajectories with 10 cycles of accelerating from a near stop and achieving speeds and decelerating were obtained. The stops were chosen to be approximately one-quarter mile apart. These trajectories were analyzed with the Comprehensive Modal Emission Model [3] (CMEM) to obtain emissions for a soak-time 60 minutes. Figure 27 shows the hydrocarbon emissions for cold-engine relative to a fully-warmed up engine and catalyst.

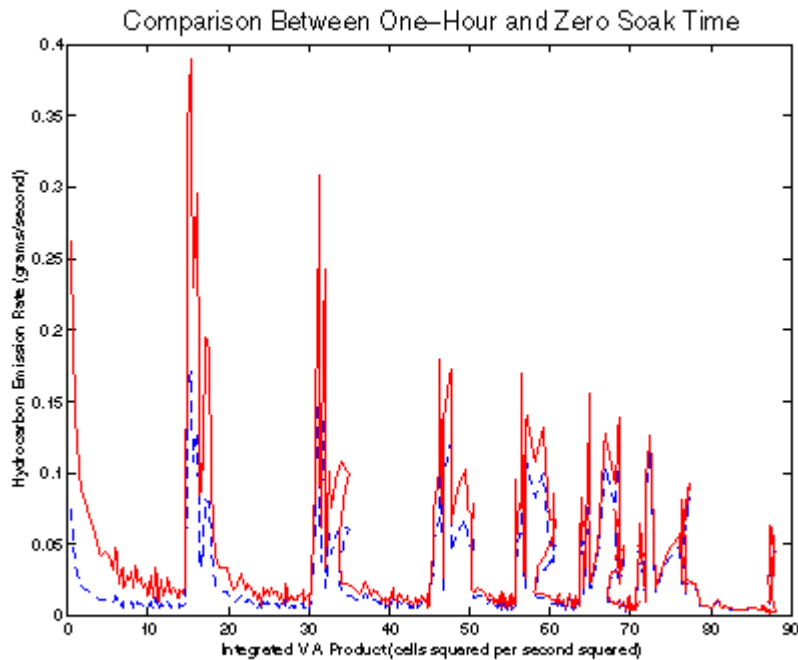


Figure 27: Emissions from a vehicle with one hour of engine off before the trajectory compared with those from a vehicle with a warm engine and catalyst.

These results were used to construct ratios for each cycle by integrating the emissions curves from the start of one peak to the beginning of the next peak. For example, by integrating from zero to 15, the cold-engine has about three times the emissions of the warm-engine. The ratios are lower for the other pollutants and the all approach one after several stop-start cycles. Figure 28 reports the calculated ratios for various pollutants.

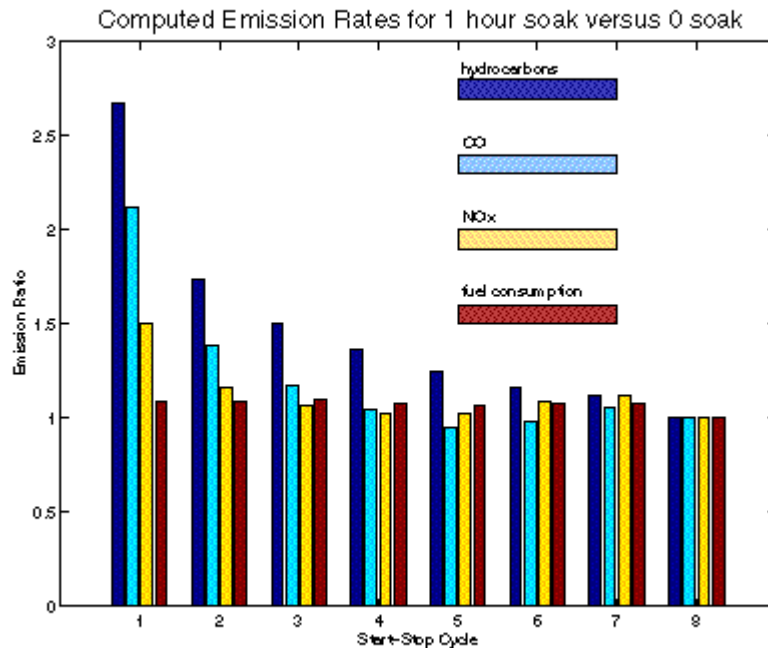


Figure 28: Ratio of cold-engine emissions to a warmed-up vehicle emissions as a function of the number of start-stop cycles since engine on.

The ratios for the eight combinations of integrated velocity-accelerator product are in the file *wcemratios*. These ratios are used to compute a multiplier that converts emissions from warmed-up vehicles into emissions for the fleet on each link

5.2.3 Fleet Dynamics

Fleet dynamics information is developed from the microsimulation cellular automata populations per speed bin and in 30-meter segment. One of the major challenges of appropriately modeling emissions is to account for the effects of different power-demands by different drivers. With light-duty vehicles, there are a wide range of accelerations available to the driver. The driver who favors harder accelerations may put the vehicle into an *enrichment* mode. During enrichment conditions, the vehicle's fuel controls switch to a fuel-rich situation that produces high emissions from the engine, and, because the catalyst is starved for oxygen, it does not reduce the emissions significantly. This fuel-control logic protects the catalyst from getting too hot, but it enormously increases the emissions of the vehicle.

Since the federal test procedure [1], was designed at a time when dynamometers were not capable of making measurements during high-power circumstances, most vehicles spend little, if any, time in the enrichment mode during the test. Thus, the vehicles can pass the EPA emission requirements even though they may have very high emissions on the highway. Although enrichment episodes occur a relatively small proportion of the time, the emissions are so much higher than normal that

they can produce a significant fraction of the total emissions. In addition, emissions of pollutants such as NO_x that are relatively unaffected by enrichment conditions are, nevertheless, quite sensitive to power levels. The upshot of all this is that it is not enough to describe the power levels demanded by a typical driver; the actual range of driving behavior must be represented. Since enrichment is expected to occur only one to a few percent of the time, the range of power levels must be described for the less than one percent of the people who drive most aggressively.

For driving on hills, enrichment becomes much more frequent, and some cars will go into enrichment while merely maintaining speed on uphill portions of major highways. In addition, highway speeds significantly higher than those encountered in the federal test procedure can also produce enrichment conditions.

There are two options to deal with this challenge:

- 1) construct a fast, accurate microsimulation that describes traffic and power demands in great detail
- 2) supplement a fast microsimulation that describes traffic properly with empirical information on power demands

One of the major difficulties with option 1 is finding adequate information describing the range of driving behavior in specific circumstances. There is much more information on traffic than there is on individual driving speeds and accelerations. For this reason, we have chosen option 2. We have developed a Traffic Microsimulator that describes traffic accurately and efficiently. From the microsimulation, we know the context in which driving occurs, and we have developed a system, the LDV Aggregate Dynamics submodule, to put empirical information into the context.

The Traffic Microsimulator output provides vehicle populations in the six speed bins in 30-meter segments (0 to 5 *cells* per second) from the start of each link in each direction. The first step is to develop a continuous distribution of speeds. We interpret the CA populations in a given segment as the integral of a continuous distribution described by the mean value and a slope term proportional to the difference between a given speed in the speed bin and the center of the speed bin. We have two constants to determine for each speed bin and each segment. The CA speed populations provide one equation for each speed bin, while the requirements that the continuous population goes to zero at the top of the highest speed bin and the population is continuous between speed bins produces the other required equations.

The next question is how is the distribution of accelerations to be determined for a given speed in a specific segment? We use two sets of empirical data to help us solve this problem. First, during EPA's three city studies [4] many vehicles were fitted with a data-logger that recorded times and speeds throughout the vehicle's travels for a significant period. This data was examined to determine what the frequency distribution of accelerations is for a given speed. More specifically, we looked for the cumulative frequency of positive accelerations. Figure 29 reports the cumulative frequency of vehicles traveling faster than one cell per second and having positive accelerations against the product of acceleration and speed.

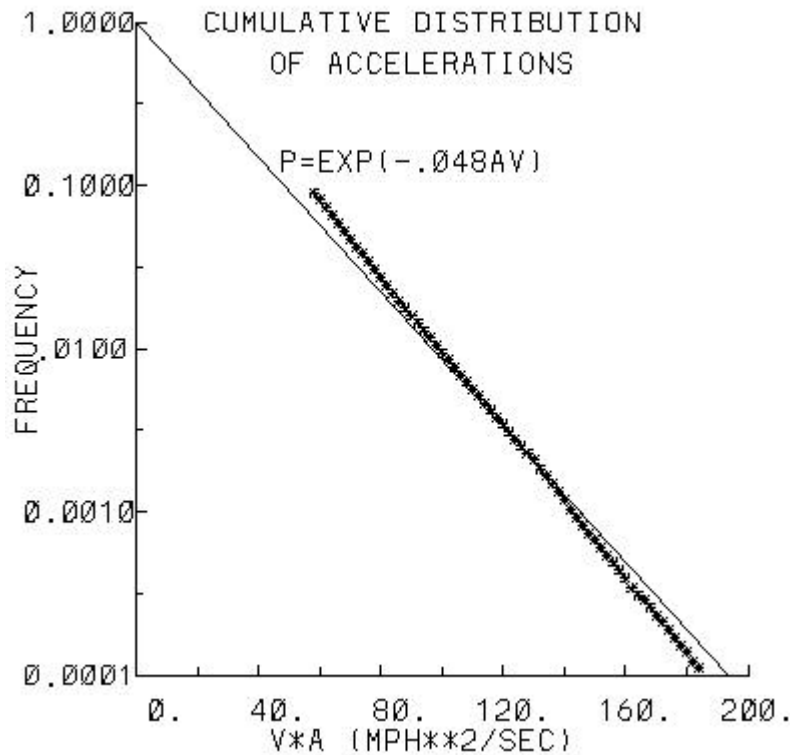


Figure 29: The cumulative distribution of accelerations from the EPA's three cities studies.

It is evident that for higher power levels the frequency of a given power level falls off exponentially with power. Similar plots can be made for speeds less than one cell per second and for decelerations. In the case of decelerations, the frequency falls off exponentially with the velocity-deceleration product. These relationships form one of the empirical underpinnings of our approach. We consider all accelerations in one of three groups:

- 1) hard accelerations
- 2) insignificant accelerations
- 3) hard decelerations

Hard accelerations are defined by accelerations greater than those associated with the 10% point on the cumulative distribution. For example, in Figure 29, the definition of hard acceleration would be velocity-acceleration products greater than 48 mph per second squared. Similar definitions can be found for hard accelerations starting at speeds less than one cell per second and for hard decelerations. We estimate the number of vehicles undergoing a hard acceleration and then choose 12 different power levels to represent different levels of aggressiveness. The power levels are chosen from the curve of Figure 29 with equal spacings in power and covering the range from a cumulative frequency of .1 to a cumulative frequency of 0.0045. The total population of vehicles undergoing hard acceleration from a given speed is then distributed over the 12 power (or equivalently acceleration) levels in accordance with Figure 29.

The next problem is to determine the fraction of the vehicles that undergo a hard acceleration in a given context. We expect that a small fraction of the vehicles will be undergoing a hard acceleration even though most of the vehicles have reached their desired driving speed. We can estimate this low level by looking at uncongested freeways. We expect that a somewhat larger fraction of the vehicles will undergo hard accelerations if they are driving on a moderately

congested freeway where they are forced to decelerate to avoid hitting other cars and then accelerate to regain their speed. If we looked at the speeds accumulated over time of such vehicles, we would find average speeds less than the desired maximum and a range of speeds from low ones to the desired maximum. Under these circumstances, the standard deviation of the speeds would be relatively large. We could estimate the frequency of hard accelerations in this case by looking at moderately congested freeways and comparing them with uncongested freeways.

We would also expect significant hard accelerations to occur when many of the vehicles are accelerating so that the average speed is increasing as we proceed along the link. For example, if we look at vehicles leaving an intersection with a stoplight, there will be vehicles going through with a green light and those accelerating from a stop caused by a red light.

A logical parameter to characterize the hard accelerations from a stoplight would be the product of acceleration and speed. Since we can express acceleration as the velocity times the rate of change of velocity with distance, we need the square of the velocity times its rate of change with distance, appropriately weighted by the number of vehicles in a spatial segment that have speed v . We can estimate the fraction of accelerations that are hard in this context by examining an arterial and looking at the vehicles leaving a stop light.

Consequently, we need empirical data on driving on uncongested freeways, moderately congested freeways, and an arterial. The second empirical underpinning for our submodule comes from data collected by the California Air Resources Board [5]. The board's contractor used a laser rangefinder to follow cars and record their speeds in a variety of circumstances. They produced seven sets of individual car trajectories organized by congestion on freeways. They also produced three sets of trajectories that they characterized as slow, medium, and fast arterials.

The data from the fastest freeway, a moderately congested freeway, and the fast arterial was used to construct a two parameter fit. We chose transformations of the acceleration-velocity product and the standard deviation of speed to the fraction of hard accelerations such that when we used the hard acceleration fractions would we get the same NO_x emissions that we would get by using the original trajectories. The LDV Aggregate Dynamics submodule carries out the continuous fit and computes the fraction of vehicles undergoing hard acceleration. When we compared the emissions from the other freeways and the two other arterials from the original trajectories with those obtained from our LDV Aggregate Dynamics submodule, we got good agreement.

The output of the LDV Aggregate Dynamics submodule is the number of vehicles in each 30-meter segment in each of 40 two-mph speed bins undergoing one of 25 different levels of acceleration. The acceleration levels include 12 hard decelerations levels, one insignificant level, and 12 hard acceleration levels. This two dimensional array is produced for each 30-meter segment of each link for each 15 minute period.

5.2.4 LDV Tailpipe Emission Submodule

We are using CMEM developed by Matt Barth [3] and his colleagues at the University of California at Riverside and the University of Michigan. Matt and his co-investigators were contracted by the National Cooperative Highway Research Program to develop an improved modal emission model for light-duty vehicles. They carried out extensive tests on over 300 vehicles chosen to represent the major types of emitters in the existing light-duty vehicle fleet. They also worked with other data to help draw associations between the tested vehicles and the fleet at large.

The model computes the tractive power by taking account engine friction losses, rolling resistance, wind resistance, changes in kinetic energy, and changes in potential energy. It also considers the power to drive accessories such as air conditioning, and it estimates drivetrain efficiency. With the engine power known, it calculates the rate of fuel consumption and engine out emissions. It treats enrichment, enleanment, and stoichiometric operations as well as cold-start operation.

Once the engine-out emissions are calculated, catalyst pass fractions are used to calculate the tailpipe emissions. The approach uses a composite vehicle to represent vehicles in the same class. A regression approach was used to define the parameters required by the model. The vehicles were all tested over cycles involving very high power demands and a variety of driving patterns.

There are composite vehicles representing normal emitting cars categorized by technology, low and high power to weight ratios, and mileages above or below 50,000. The technology categories are:

- 1) no catalyst
- 2) two-way catalyst
- 3) three-way catalyst with carburetion
- 4) three-way catalyst with fuel injection
- 5) Tier 1

Only the last two technologies are broken into mileage or power-to-weight ratio groupings. There are high emitting composite vehicles for technologies 3) through 5), but they are not further subdivided into power-to-weight ratios or mileage groupings. There are composite vehicles representing normal-emitting trucks with model year categories:

- 1) pre-1979
- 2) 1979 to 1983
- 3) 1984 to 1987
- 4) 1988 to 1993
- 5) 1994 and newer

In age categories 1) through 3,) there is only a single composite vehicle. For age category 4), there are categories for trucks above and trucks below 3750 pounds loaded vehicle weight, while for category 5) there is a category for trucks with loaded vehicle weights between 3751 and 5750 pounds and a category for gross vehicle weights between 6001 and 8500 pounds. There are composite vehicles representing high-emitting trucks for model years 1984 to 1987, 1988 to 1993, and 1994 and newer. In the high-emitting category, there are no breakdowns by vehicle weight.

The relationships between speed and acceleration produced by CMEM are embodied in the file *array.out*. This file gives the composite vehicle emissions for 2 mph speed bins and 1.5 foot per second acceleration bins. The array was developed by using CMEM for 1 second trajectories starting in the specified speed bin and accelerations associated with the chosen acceleration bin. This form of the model does not treat longer-term history effects where the emissions are dependent upon the specific speed-acceleration patterns leading up to the second being modeled. The model does not currently treat grade effects, although that is a feature that will be included in future versions. CMEM does have grade effects; recomputing *array.out* with the average grade for the link would give an approximation of grade effects.

5.3 Algorithm

The first step in estimating the velocity-acceleration distribution is to make a continuous fit to the densities (number of vehicles per unit speed and per unit space). A simple fit that is of the form:

$$d_{ij}(\delta v) = f_{ij} + h_{ij}\delta v$$

where,

$$\delta v = v - j\Delta$$

and i represents the spatial cell and j represents the speed bin with Δ being the cell width, 7.5 meters or 7.5 meters per second for the speed bins. The constant term f_{ij} is given by:

$$f_{ij} = N_{ij} / (4\Delta^2)$$

because the length of the box is 4Δ , and the width in velocity space is Δ . The gradients slope terms, h_{ij} are found by setting d_{ij} to zero at the top of the highest speed bin and solving for h_{ij} . Continuity relationships are used to determine the h_{ij} 's for the slower speed bins. However, this procedure may lead to negative densities over a portion of a speed bin. This problem is solved in one of two ways. First, if the negative value occurs in the slowest of the speed bins that have vehicles, the density relationship is assumed to hold down to a value of dv denoted dv_i where the density falls to zero and remains there. In other words, in the slowest, populated speed-bin the distribution extends from $dv=(\Delta/2)$ to $dv=dv_i$ rather than to $dv=-\Delta/2$. The second situation occurs where the potential negative values occur in an intermediate speed-bin. In this case, the continuity condition at the speed-bin boundaries is relaxed and h_{ij} is set to zero.

Once the densities are known, the various moments of speed are calculated. Specifically, the zeroth-moment (average density), first moment (flux), second moment (needed for speed variance), and the third moment, whose gradient is related to power are calculated. The flux is divided into thirds and used to calculate the breakpoints between the slowest one-third ($j=n_{13}$ and $dv=dv_l$), the middle one-third, and the fastest one-third ($j=n_{23}$ and $dv=dv_h$). Once the breakpoints are determined, the third-moments of speed are calculated for each third of the flux. The probability of a hard-acceleration is then estimated from:

$$P_a = \max(P_\sigma, P_\varphi)$$

with

$$P_\sigma = P_0 + P_{\sigma 1} v_{avg}^2 (\sigma - \sigma_r)$$

where s is the standard deviation of speed, v_{avg} is the average speed, and s_r is the standard deviation of speed for an uncongested freeway. The calibration constants, p_0 and P_{s1} are determined separately for each third of the flux (slowest, intermediate, and fastest), but the standard deviation and the average speed refer to the entire vehicle flow in the segment. The foregoing treatment associated with the standard deviation of speed dominates when the average speed along the link is not changing or is changing very slowly.

When there are significant speed changes along the link, the second component, P_{sp} given by:

$$P_{sp} = P_{sp0} + P_{sp1}(sp - 0.02)$$

with the speed gradient parameter defined by:

$$sp = \frac{\text{gradient-of-third-moment}}{\text{zeroth-moment} \times \Delta^2}$$

dominates. In this case, the moments and the calibration constants are calculated for each third of the flux. The calibration constants were chosen based on maximum gradients for vehicles leaving a signal on a fast arterial.

The code also accounts for vehicles that do not reach the end of link. It is assumed that these vehicles fall into the slowest third of the vehicles and sufficient flux among the slowest third of the vehicles in the segment upstream of the segment under concern is added to the downstream segment before the gradient is calculated. This procedure requires that the cutoff points between the various thirds of the flux be recalculated for the downstream segment. The underlying assumption for this procedure is that the flux should be approximately constant if all vehicles complete the link.

5.4 Usage

The Emissions Estimator is designed to produce emissions appropriate for simulations of air quality over a metropolitan area as would be done with EPA's MODELS-3. It does not produce intersection emissions in this version. It is designed to produce fleet average emissions rather than emissions from individual vehicles. It uses continuous approximations for densities and thus requires that many vehicles be considered over at least a 15-minute period in order for appropriate statistics to be developed.

In this version, it does not consider grades nor different fleet mixes. In future versions, different fleet mixes and grades will be treated by producing versions of the input file *array.out* appropriate to the link under consideration. This version does not calculate the probability of hard decelerations and thus does not properly treat hydrocarbon emissions from rapidly slowing vehicles. Pilot studies suggest that a very similar treatment to that used for hard accelerations can be used to treat hard decelerations. In addition, on-ramps are not treated properly in this version of the code. Freeway on-ramps appear to have a somewhat different speed-acceleration distribution from that used in the model development to date. Once again, pilot studies suggest that a similar formulation with different parameters could be used to treat freeway on-ramp emissions.

The current version of the model considers only one category of soak-time. Future versions will consider three representative soak-times.

There are a number of assumptions in the current version of the code. First, the number of 30-meter segments on a link populated with vehicles must be at least three, all of which must have vehicles. There can be empty cells before and after a set of populated ones, but there must not be any unpopulated segments interspersed with populated ones. Furthermore the speed histograms

should be constructed from one second sampling, summed over 15 minutes or more. All links must be one-way.

Table 31 below lists the TRANSIMS configuration file keys that are specific to the Emissions Estimator.

Table 31. Configuration keys specific to the Emissions Estimator.

Configuration Key	Description
EMISSIONS_ARRAY_PARAMETERS_FILE	file name for the <i>ARRAY.INP</i> file which contains the parameters describing the number of records and increments used in the <i>array.out</i> file
EMISSIONS_COMPOSITE_INPUT_FILE	file name for the <i>array.out</i> file which contains the composite vehicle emissions in 2-mph speed bins and 1.5 feet/second acceleration bins
EMISSIONS_MICROSIM_VELOCITY_FILE	file name for the <i>readca.out</i> file produced by <i>Readca</i> which contains the reformatted microsimulation velocity summary data
EMISSIONS_VEHICLE_COLD_DISTRIBUTION	file name for the <i>vehcold.dis</i> file which contains the distribution of vehicles entering the link stratified by the time integrated, velocity-acceleration product (power) and by soak time
EMISSIONS_WCEM_RATIOS_FILE	file name for the <i>wcemratios</i> file which contains the ratios of cold emissions to hot engine emissions.

Table 32 below lists the TRANSIMS configuration file keys that must be set to a specific value for the Emissions Estimator to calculate emissions correctly.

Table 32. Configuration keys that must be set to a specific value for the Emissions Estimator.

Configuration Key	Description	Value
OUT_SUMMARY_BOX_LENGTH_n	Length of the summary boxes (in meters)	This value needs to be set to 30
OUT_SUMMARY_SAMPLE_TIME_n	Frequency (in seconds) at which to accumulate data	This value needs to be set to 1
OUT_SUMMARY_TIME_STEP_n	Frequency (in seconds) at which to report data	This value needs to be set to 900
OUT_SUMMARY_VELOCITY_BINS_n	Number of bins used to cover the range of the velocity histogram.	This value needs to be set to 6

5.5 Tutorial

Before running the Emissions Estimator, the Traffic Microsimulator must be run to collect velocity summary data. Refer to Volume 3—*Files*, Sections 9.5 and 9.6, for configuration keys that must be set to specific values.

☛ The Emissions Estimator works on velocity summary data collected on the Tee Network only in this release.

Run *Readca* in order to convert the microsimulation velocity summary data file into a format that can be read into the Emissions Estimator.

```
% Readca <velocityFilename>
```

The file *readca.out* is produced.

Run the Emissions Estimator to produce a file of emissions that can be read into the Output Visualizer. The configuration file used to run the Traffic Microsimulator to collect velocity data must be specified on the command line. Before running this program, verify that all of the necessary input files are present in the directory you will be running from or that all the Emissions Estimator configuration keys are set properly. These include *ARRAY.INP*, *array.out*, *vehcold.dis*, *wcemratios* files, and the *readca.out* file produced above. Refer to Volume 3—*Files*, Section 9, for detailed explanations of the various input and output files.

```
% EmissionsEstimator <configFilename>
```

The files *readart.out* and *emissions.out* will be produced.

Run the Output Visualizer to display the emissions. The Output Visualizer needs the configuration file used to run the microsimulation in order to know which network to bring up. Refer to Volume 2—*Software*, Part 1—*Modules*, Section 6, for help running the Output Visualizer.

```
% Vis <configFilename>
```

Within the Output Visualizer,

- select **File→Open Network**
- select **File→Open Variable Size Box Data**
- choose the *emissions.out* file produced by the Emissions Estimator

To cycle through the different emissions columns (VTT, NOX, CO, HC, FE, and FLUX), select **View→Emissions**. The type of data being display is labeled in the lower window next to the time. The threshold values and colors are in Example J of Volume 3—*Files*, Section 9.6.

To animate the emissions data through different timesteps select **Modes→Animation**. The Animation tools become available at the bottom of the Output Visualizer's window.

To display three-dimensional emissions where not only the color of the summary box but also the height depends on the emissions value.

- rotate the network so the Z is at about 50 and the X to about 300

- select **Modes→Lights** to get better shading
- select **View→3D Plans**
- click [OK]

The network will look funny to begin with. Just start cycling through the emissions data fields (**View→Emissions**) to correct the display. The three-dimensional emissions can also be animated.

Refer to Volume 3—*Files*, Section 9.6, for an example of what these emissions look like in 3-D.

5.6 Troubleshooting

5.6.1 Messages and ERRORS from *Readca*

Usage: Readca <velocityFilename>

Readca reads in a Microsimulation Velocity Summary output file, processes the data, and then writes out a *readca.out* file for input into the Emissions codes. This message occurs when the velocity file is not specified when running the Readca program. To run *Readca* you must specify the velocity file on the command line.

ERROR: Unable to open input file: <velocityFilename>

This message occurs when the *Readca* program is either unable to locate or unable to open the input file specified on the command line. Make sure that you have the correct location for your input file.

ERROR: Unable to open output file: readca.out

This message occurs when the *Readca* program is unable to open the output file *readca.out*. This may occur due to lack of disk space or lack of write privileges in the directory in which you are attempting to run *Readca*.

ERROR: Bad header for velocity file

This message occurs when the velocity summary input file is empty.

ERROR: Bad header for velocity file: field COUNT5 is missing

This message, and other similar messages with different field names, occurs when there is a missing necessary field in the velocity file. The fields needed are TIME, LINK, NODE, DISTANCE, COUNT0, COUNT1, COUNT2, COUNT3, COUNT4, and COUNT5. If any of these fields are missing, the simulation must be rerun with the configuration modified to allow collection of all of these fields.

ERROR: Exceeded maximum number of boxes for a link (120 boxes -> 3600 meters)

This message occurs when the velocity file contains data for links that are longer than 3600 meters long. Either remove these elements from your data file or rerun the simulation and exclude collection on links greater than 3600 meters.

5.6.2 Messages and ERRORS from the *EmissionsEstimator*

Usage: EmissionsEstimator <config file>

This message occurs when the *EmissionsEstimator* is run without a configuration file as an argument on the command line. Specify the configuration file used to run the Traffic Microsimulator to produce the velocity data for the Emissions Estimator.

ERROR: Unable to open the CA input file: readca.out

This message occurs when the *EmissionsEstimator* program is unable to locate or open the input file *readca.out*. Make sure that the file is present in the directory in which you are attempting to run the *EmissionsEstimator* or that the configuration key EMISSIONS_MICROSIM_VELOCITY_FILE contains the correct file name.

ERROR: Unable to open the array input file: array.out

This message occurs when the *EmissionsEstimator* program is unable to locate or open the input file *array.out*. Make sure that the file is present in the directory in which you are attempting to run the *EmissionsEstimator* or that the configuration key EMISSIONS_COMPOSITE_INPUT_FILE contains the correct file name.

ERROR: Unable to open the array INP input file: ARRAY.INP

This message occurs when the *EmissionsEstimator* program is unable to locate or open the input file *ARRAY.INP*. Make sure that the file is present in the directory in which you are attempting to run the *EmissionsEstimator* or that the configuration key EMISSIONS_ARRAY_PARAMETERS_FILE contains the correct file name.

ERROR: Unable to open the vehicle input file: vehcold.dis

This message occurs when the *EmissionsEstimator* program is unable to locate or open the input file *vehcold.dis*. Make sure that the file is present in the directory in which you are attempting to run the *EmissionsEstimator* or that the configuration key EMISSIONS_VEHICLE_COLD_DISTRIBUTION contains the correct file name.

ERROR: Unable to open the ratios input file: wcemratios

This message occurs when the *EmissionsEstimator* program is unable to locate or open the input file *wcemratios*. Make sure that the file is present in the directory in which you are attempting to run the *EmissionsEstimator* or that the configuration key EMISSIONS_WCEM_RATIOS_FILE contains the correct file name.

ERROR: Unable to open the output file: emissions.out

This message occurs when the *EmissionsEstimator* program is unable to open the output file *emissions.out*. This may occur due to lack of disk space or lack of write privileges in the directory in which you are attempting to run the *EmissionsEstimator*.

ERROR: Unable to open the output file: readart.out

This message occurs when the *EmissionsEstimator* program is unable to open the output file *readart.out*. This may occur due to lack of disk space or lack of write privileges in the directory in which you are attempting to run the *EmissionsEstimator*.

ERROR: Input file vehcold.dis does not have enough soak times

This message occurs when eight lines are not present in the *vehcold.dis* file. There must be a line for the number of soak time bins. Refer to Volume 3—*Files*, Section 9.5, for an example of this file.

ERROR: Input file wcemratios does not have enough data

This message occurs when four columns and eight rows are not present in the *wcemratios* file. Refer to Volume 3—*Files*, Section 9.5, for an example of this file.

ERROR: Input file ARRAY.INP does not have enough data

This message occurs when ten data elements are not present in the *ARRAY.INP* file. Refer to Volume 3—*Files*, Section 9.5, for an example of this file.

ERROR: Missing a header line from input file: array.out

This message occurs when the *array.out* file does not have at least two lines in the file. Refer to Volume 3—*Files*, Section 9.5, for an example of this file.

ERROR: Out of input data in file: array.out

This message occurs when there is not enough data in the *array.out* file. It must contain six columns of data. The number of rows of data must be the maximum number of acceleration bins multiplied by the maximum number of velocity bins. These two values are read in from the *ARRAY.INP* file. In this release of the emissions modules, these values are 17 acceleration bins and 40 velocity bins. There should be 680 rows of data in the *array.out* file.

ERROR: *Exceeded maximum number of boxes for a link (120 boxes -> 3600 meters)*

This message occurs when the *readca.out* file contains data for links that are longer than 3600 meters long. Either remove these elements from your data file or rerun the simulation and exclude collection on links greater than 3600 meters. This message should not occur because it should be caught in a run of the *Readca* program.

ERROR: *Missing data from input file: readca.out*

This message occurs when the *readca.out* file is missing data. The value after *nv =* should be the number of lines of data following for that particular time, link, and node. Errors in the *readca.out* file would have been introduced by the *Readca* program.

ERROR: *Data set contains a link with empty interior boxes*

There must be vehicles present in a consecutive set of boxes on any particular link. Links may contain boxes that are empty at the beginning and the end of the link. In order to calculate emissions correctly, boxes that contain vehicles must all be consecutive. There may not be empty boxes in the middle of a set of boxes with vehicles. One way to check if a box contains vehicles is to look across a particular row of data in the *readca.out* file. If all values are zero, then there were no vehicles in that box. If this situation occurs in your data, you can remove the data for those links from your *readca.out* file or rerun the Traffic Microsimulator to exclude data collection on those particular links.

ERROR: *NaN or Inf found in one of the output files (readart.out and/or emissions.out)*

This message occurs when there is a calculation error in the *EmissionsEstimator* program (such as a divide by zero). No known fix for this.

5.7 References

1. FTP.(1989). Code of Federal Regulations. Title 40. Parts 86-99 (portion of CFR which contains the Federal Test Procedure), Office of the Federal Register.
2. Novak, J. H., R. L. Dennis, D. W. Byun, J. E. Pleim, K. J. Galluppi, C. J. Coats, S. Chall, and M. A. Vouk, "EPA Third-Generation Air Quality Modeling System: Volume 1 Concept", EPA600/R95/082, 1995, US Environmental Protection Agency, Research Triangle Park, NC 27711.
3. Barth, M., T. Younglove, T. Wenzel, G. Scora, F. An, M. Ross, and J. Norbeck (1997), "Analysis of Modal Emissions for a Diverse in-use Vehicle Fleet." Transportation Research Record, No. 1587, Transportation Research Board, National Academy of Science, pp. 73-84.
4. USEPA (1993), "Federal Test Procedure Review Project: Preliminary Technical Report," EPA 420-R-93-007, 1993, Office of Air and Radiation, Washington, DC.
5. Effa, Robert C. and Lawrence C. Larson, "Development of Real-World Driving Cycles for Estimating Facility-Specific Emissions from Light-Duty Vehicles," California Environmental Protection Agency - Air Resources Board - presented at the Air and Waste Management Association Specialty Conference on Emission Inventory 1993, Pasadena, California.

6. OUTPUT VISUALIZER MODULE

6.1 Overview

The TRANSIMS Output Visualizer module allows users to display various input and output data sets and provides tools to facilitate the analysis of the data sets. Among the types of data the Output Visualizer displays/analyzes are:

- Plans – Single aggregated or filtered overlaid on a given network.
- Vehicles – Can be colored by velocity, type, etc. and animated on a given network.
- Summary Data – Densities and average velocities are drawn and animated on constant length boxes.
- Variable Size Box Data – Any user selectable data value can be shown drawn on any link of any size on a given network. This makes it possible to display data of vastly differing types from emissions levels to plans.

6.2 Requirements

The Output Visualizer currently runs under Sun Solaris and Linux operating systems. Hardware requirements include only a 3-D capable graphics board such as a Sun Creator 3D or better for Sun workstations or an OpenGL-compatible graphics board for systems running Linux. OpenGL-compatible graphics boards for Intel/Linux systems add very little to the cost of machines; in fact, most laptops will run the Output Visualizer without the addition of a graphics board upgrade. A three-button mouse is also required. Software requirements include:

- OpenGL or Mesa3D - graphics library
- GLUT - a multiplatform windowing system library.

6.3 Tutorial

6.3.1 Run the Output Visualizer

Run the Output Visualizer by executing the following command line

```
Vis configfile
```

where `Vis` is the executable file, and `configfile` is the name of the configuration file upon which you would like to view your data; Table 33 lists the required entries for the configuration file. Do not run this application in the background because you will need a text output window for retrieving various types of information. The Output Visualizer will be displayed as shown in Figure 30.

Table 33. Required Output Visualizer configuration keys.

Configuration Key	Description
NET_DIRECTORY	must be set to directory where network tables are
NET_NODE_TABLE	must be set to the name of the node table
NET_LINK_MEDIAN_HALFWIDTH	must be set to $\frac{1}{2}$ of GBL_LANE_WIDTH. It is the distance that the links are offset from the node. (Note: this key must be the same for collecting output and running the Output Visualizer; otherwise, vehicles will not be centered properly in lanes.)
NET_LINK_TABLE	must be set to the name of the link table
NET_POCKET_LANE_TABLE	must be set to the name of the pocket lane table or to an empty pocket lane table
NET_PARKING_TABLE	must be set to the name of the parking table or to an empty parking table
NET_TRANSIT_STOP_TABLE	must be set to the name of the transit stop table or to an empty transit stop table
GBL_CELL_LENGTH	The length of a cell in meters. Default is 7.5
GBL_LANE_WIDTH	The width of a lane in meters. Default is 3.5 (Note: The settings for GBL_LANE_WIDTH used by the Output Visualizer must be the same as used by the output system in order for the vehicles to be placed properly on the network.)
OUT_SNAPSHOT_SUPPRESS_1	These keys determine what fields to suppress in the snapshot output file. It is much faster to use the <code>vehtobin</code> utility to convert the snapshot files to binary files ready to be input to the Output Visualizer. In this case, this key should be set to: ACCELER; DRIVER; USER; LANE; NODE; DISTANCE If you prefer to use the <code>convcars</code> utility to convert the snapshot files, this key should be set to: ACCELER; AZIMUTH; DRIVER; EASTING; ELEVATION; NORTHING; USER In general, <code>vehtobin</code> performs no calculations when converting the output whereas <code>convcars</code> calculates the vehicle positions from link, node, lane and distance positions so longer conversion times are necessary.
VIS_SINGLE_BUFFERED	If it is non-zero, the Output Visualizer will run in single buffered mode. Double buffering should be used if it is at all possible (virtually all situations).
VIS_BOX_LENGTH	Should be 150 (meters). This parameter is the summary box length in meters.

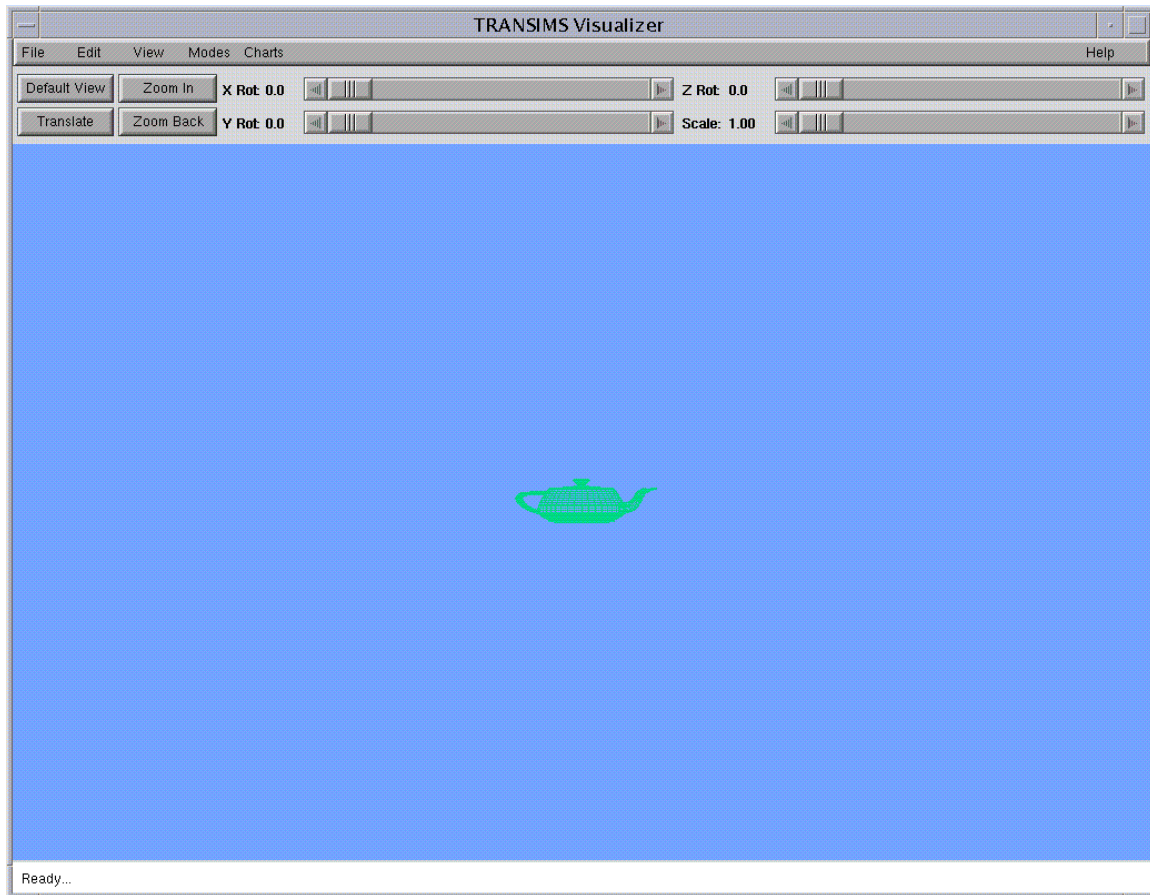


Figure 30: TRANSIMS Output Visualizer as it appears initially.

The TRANSIMS Output Visualizer graphical user interface allows you to manipulate three-dimensional objects. The toolbar consists of buttons and sliders to achieve this purpose. The sliders allow for rotating the object about any axis and for scaling the object about the current center of the display.

- The Default View button resets the viewing transformations to their default values.
- The Translate button allows you to drag the graphical objects to a new location by first clicking on the button and then clicking on any point in the viewing area and dragging to another point and releasing the left mouse button.
- The Zoom In button allows you to magnify the display of a selected area by first clicking on the button and then clicking and dragging with the left mouse button on the display area that you would like to see in more detail.
- The Zoom Back button resets the viewing transformations to those that existed prior to the last zoom in command.

The bottom edge of the display is used to display the current status and/or timestep and variable being displayed. The menu system consists of six pulldown menus outlined below.

- 1) **File** – allows for the open and closing of data files and networks

- 2) **Edit** – allows for the finding of objects, labeling, and changing the background color.
- 3) **View** – allows for the selection of what type of data and/or style the data should be displayed.
- 4) **Modes** – allows for the selection of various viewing modes such as whether to use the lighting model, overlay mode, 3-D or 2-D network, etc..
- 5) **Charts** – allows for the production of various types of graphs.
- 6) **Help** – allows access to the help facility.

Initially, the Output Visualizer displays a wire frame teapot, read in the network described in the configuration file by selecting the **File→Open Network** menu option. The text in the status bar changes to *Reading in the Network...* while the network is being read and returns to *Ready...* once completed. The Output Visualizer displays the network centered in the viewing window with the correct aspect ratio as shown in Figure 31.

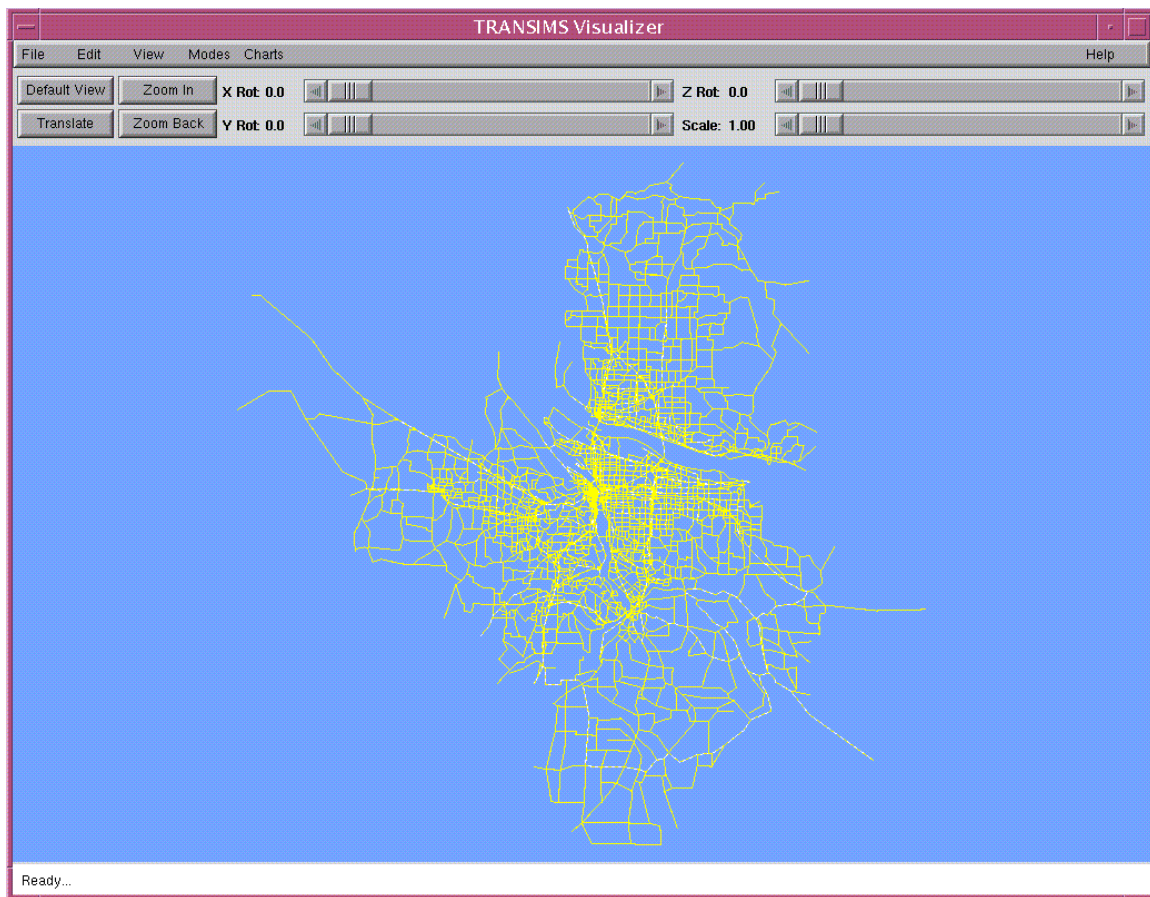


Figure 31: The Output Visualizer displaying the Portland EMME/2 Network.

6.3.2 Practice Manipulating the View

6.3.2.1 Translating

Click [**Translate**], then click on a point in the viewing area with the left mouse button and hold it down, drag to another point in the viewing area and release the left mouse button. The Output Visualizer redraws the network translated by the distance between the two points. This translation can be undone by clicking [**Default View**], which resets the viewing transformations to their original values and redraws the network.

6.3.2.2 Zooming

Click [**Zoom In**], then click on a point in the viewing area with the left mouse button and hold it down, drag to another point in the viewing area and release the left mouse button. The Output Visualizer displays the area in the network you selected with the maximum magnification that will fit entirely within the current window. This zoom operation can be undone by clicking [**Default View**], which resets the viewing transformations to their original values and redraws the network. It can also be undone by clicking [**Zoom Back**].

Currently, [**Zoom Back**] only reverses the last zoom in operation, it does not reverse multiple zooming operations. Note that when selecting an area to zoom into, any diagonal in any direction will work. For example, the same results can be produced by clicking on the upper left then dragging to the lower right of an area as when clicking on the lower right and dragging to the upper left. A sample zoomed in view of the network is shown in Figure 32.

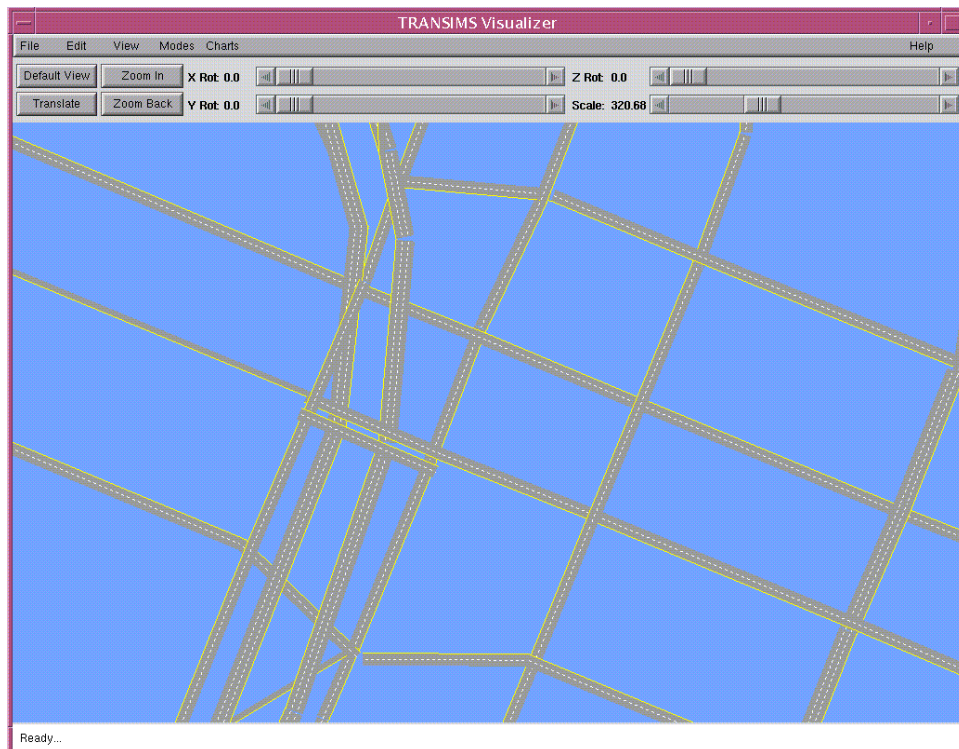


Figure 32: The Output Visualizer displaying a zoomed-in view of the Portland EMME/2 Network.

Note that lane dividers are displayed as white dashed lines, and the yellow lines are drawn on the left side of links thereby indicating the direction of traffic.

6.3.2.3 Rotating

Drag the thumb button to the right of the “Z Rot:” label to the right. The entire network is rotated about the Z-axis by the amount shown by the slider. You may also click on the arrows at the ends of the slider to increment or decrement the rotation.

6.3.2.4 Scaling

Drag the thumb button to the right of the “Scale:” label to the right. The entire network is magnified about the current center of the window by the amount shown by the slider. You may also click on the arrows at the ends of the slider to increment or decrement the scale.

6.3.2.5 Displaying Vehicle Evolution Files

Vehicle Evolution Files are created by converting vehicle snapshot files into a binary format with the *veh Tobin* utility (refer to Section 6.4.2 for details). Read in a Vehicle Evolution file by selecting **File→Open Vehicles**. The text in the status bar changes to *Reading in Vehicles...* while the vehicle evolution file is being read and will return to *Ready...* once completed. The Output Visualizer displays the vehicles as points when zoomed out and transitions to displaying vehicles as triangles when zoomed in. The display should now look similar to that shown in Figure 33.

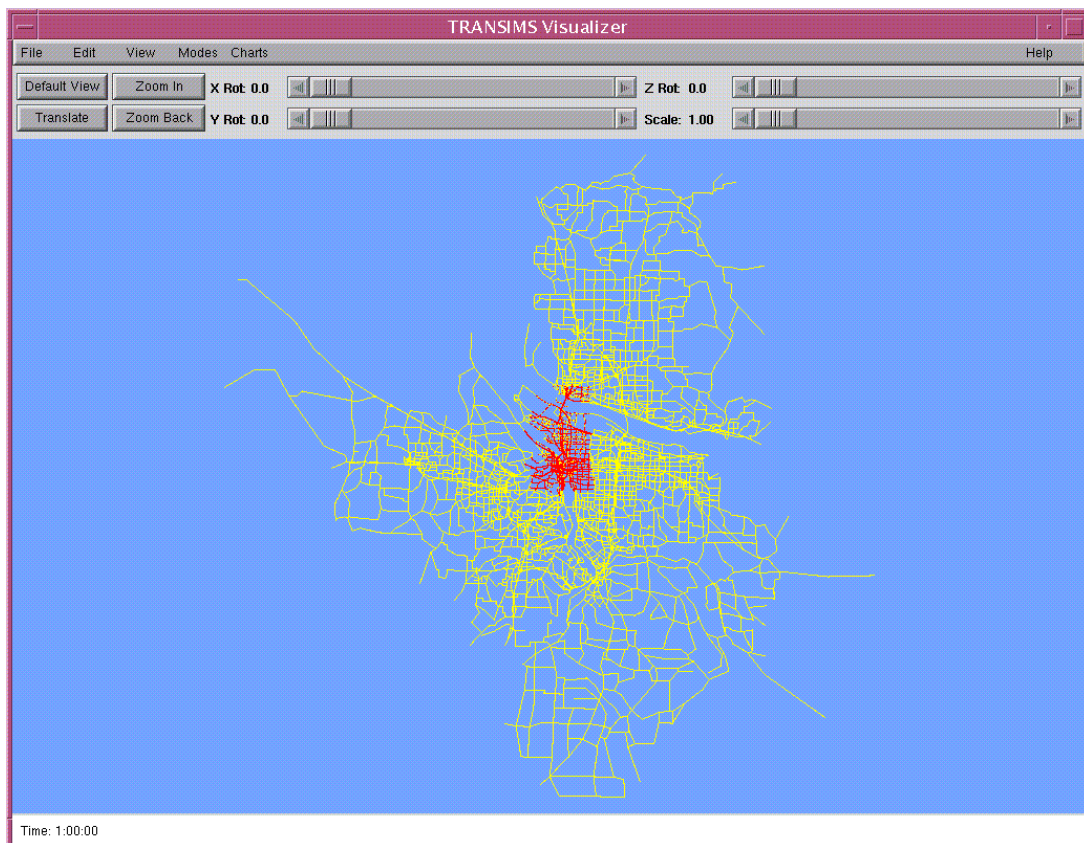


Figure 33: The Output Visualizer displaying vehicles on the Portland EMME/2 Network.

6.3.2.6 Looking for High Vehicle Densities

Click on **Modes**→**Animation** and sliders and buttons will be displayed on the status bar at the bottom of the window. The leftmost slider is labeled “Thresh:” and it controls the transparency of the vehicles when vehicles are shown as points. When Thresh is 1.00, the vehicles are not transparent, and as thresh declines they become more transparent. Thus, since in very zoomed out views vehicles are smaller than pixels, pixels with multiple transparent vehicles will appear darker. Therefore, the highest densities of vehicles will appear as the darkest pixels as the Thresh slider goes towards zero. An example is shown in Figure 34.

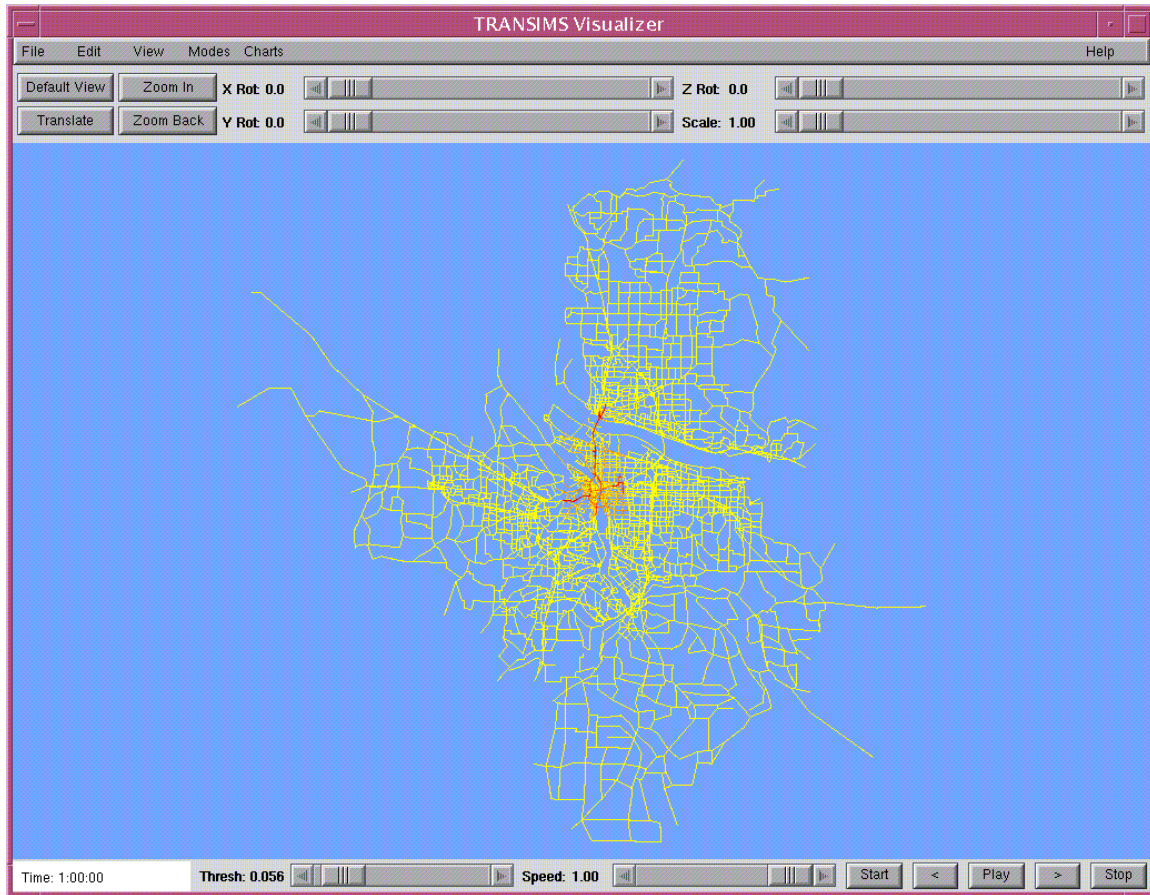


Figure 34: The Output Visualizer displaying high vehicle densities on the Portland EMME/2 Network.

Zoom in and the vehicles are displayed as triangles as shown in Figure 35. The point of the triangle in the middle of the lane is the front of the vehicle.

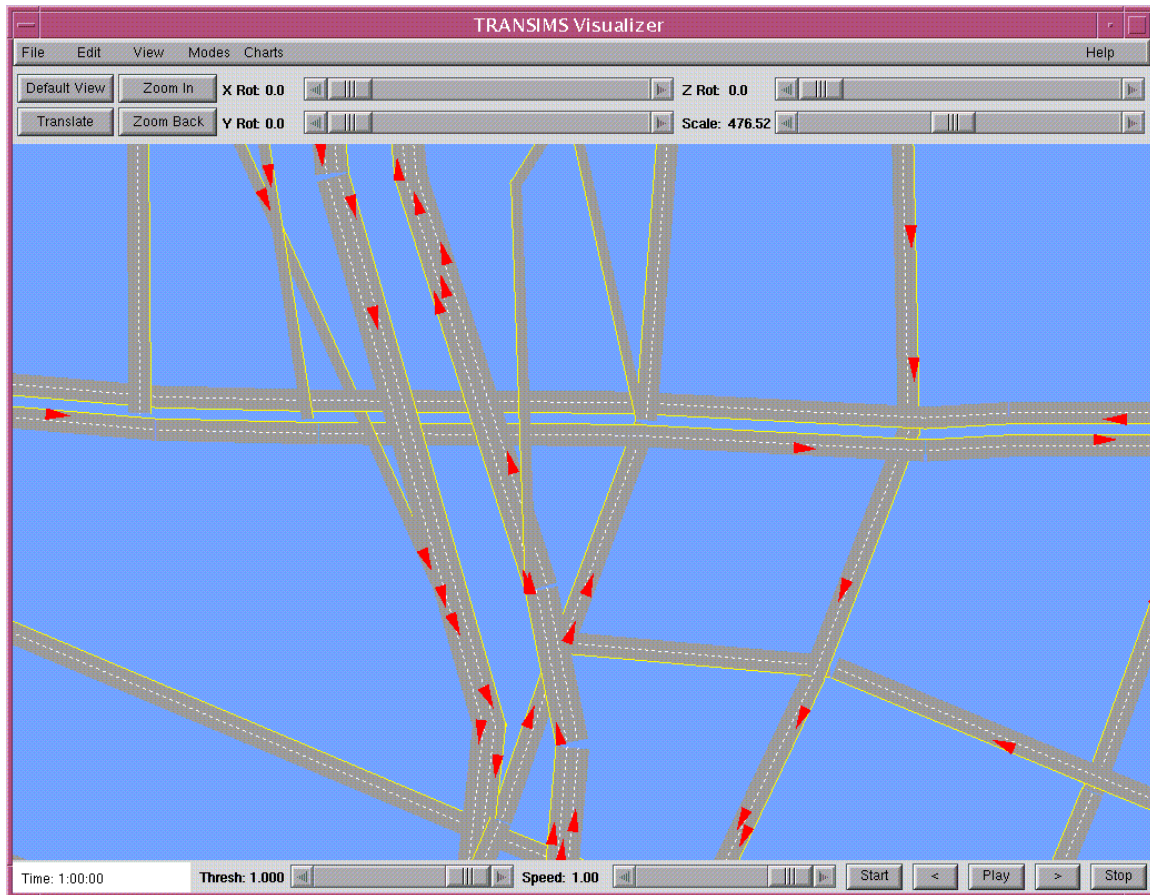


Figure 35: The Output Visualizer displaying vehicles on the Portland EMME/2 Network in a zoomed-in view.

6.3.2.7 Coloring the Vehicles by Different Schemes

Select **View**→**Vehicles** and the Vehicle Viewing Parameters dialog box is displayed. This dialog box allows you to select a scheme by which to color the vehicles. You may have the vehicles all appear in one color; color the vehicles by type, number of passengers, or velocity; or color the vehicles with random colors. Coloring the cars randomly is useful in watching the flow of large numbers of vehicles while still being able to track a single vehicle. Different vehicle types can be distinguished by coloring by Type. Vehicles can also be drawn in 3-D in any mode by selecting the 3-D Vehicles checkbox at the bottom of the Vehicle Viewing Parameters dialog box.

Select the Color by Velocity Mode and click OK. The vehicles are colored according to velocity, dark red being the slowest, then bright red, then blue, then bright green, followed by dark green for the fastest vehicles.

6.3.2.8 Animating the Vehicles

Click the left mouse button on the viewing area to start the animation. As each frame of the animation is drawn, the timestep is written in the status bar. Click on the right mouse button to stop the animation. Alternatively, you can also click **[Play]** to start the animation and **[Stop]** to stop the animation. Dragging the Speed slider towards zero (left) slows the animation

speed. The starting timestep can be displayed by clicking [**Start**]. You can advance the timestep by clicking the [>] or, you can decrease the displayed timestep by clicking [<].

6.3.2.9 Animating Faster

Since the entire network is being redrawn for each frame, the animation may appear quite slow, particularly for networks with large numbers of links. The animation speed can be greatly increased by switching the Overlay mode to on. This can be done by selecting **Modes→Overlay**. Set the overlay mode on, and restart the animation. The animation frame rate should now be much faster since the network is drawn only once and is copied from memory into the viewing area for each subsequent timestep. Stop the animation and free the memory used for the vehicle evolution data by selecting **File→Close Vehicles**.

6.3.2.10 Retrieving Information about a Vehicle

Clicking on the front middle of a vehicle (when they are large enough to be represented as triangles) with the MIDDLE mouse button returns information about the vehicle. When you have clicked on a point within 2.5 meters of the middle front of a vehicle, information is printed in the text window from which you started the Output Visualizer. A typical output would be:

Found Vehicle: ID 336565 Type 16 Passengers 0 Velocity 24.59 Link ID 16155

Since the first vehicle whose point is within the 2.5 meter tolerance is selected for output, try to select a point where it is not likely that the point will be within the tolerances of another vehicle. If a vehicle cannot be found at the point where you have clicked, the following message is displayed:

Vehicle NOT FOUND at xcoordinate ycoordinate

6.3.2.11 Viewing Variable Size Box Data

Read in a Variable Size box data file by selecting **File→Open Variable Size Box**. The text in the status bar will change to *Reading in Variable Size Boxes...* while the file is being read and will return to *Ready...* once completed. The Output Visualizer displays the first column of data in the file by default. Links are colored according to the user selectable colormap. Other columns can be displayed by selecting **View→Next Data Column**, which increments the data column being displayed. Animation of the data can be displayed by clicking the left mouse button in the viewing area; stopping the animation is done by clicking the right mouse button.

Variable Size box data files can be produced from plans by using the Plan2BoxSummary utility (see Section 6.5.3). The emissions package also produces files of this type.

6.3.2.12 Viewing the Data in 3-D

Since it can be very difficult to identify outlying values with colormapped data, 3-D capabilities are also provided. It is much easier to identify outlying data when viewing tall bar graphs than by coloring the data values with a given colormap. The data can be viewed in 3-D by selecting **View→3D Plans**. A dialog box is displayed that allows you to enter a scale factor for multiplying the data values into 3-D bar heights. Enter an appropriate scale factor, and click [**OK**]. The display is rendered with 3-D bars on the boxes. Rotate the display by approximately 290 degrees with the X Rot slider. Animate the data by left clicking. Stop the animation by right clicking. The

display may look confusing at this point because all faces of the 3-D bar boxes are colored exactly the same. You will not be able to distinguish the tops of the boxes from the sides. The lighting model fixes this problem. Switch on the lighting model by selecting **Modes→Lights**. A sample view of 3-D Variable Size Box data is shown in Figure 36.

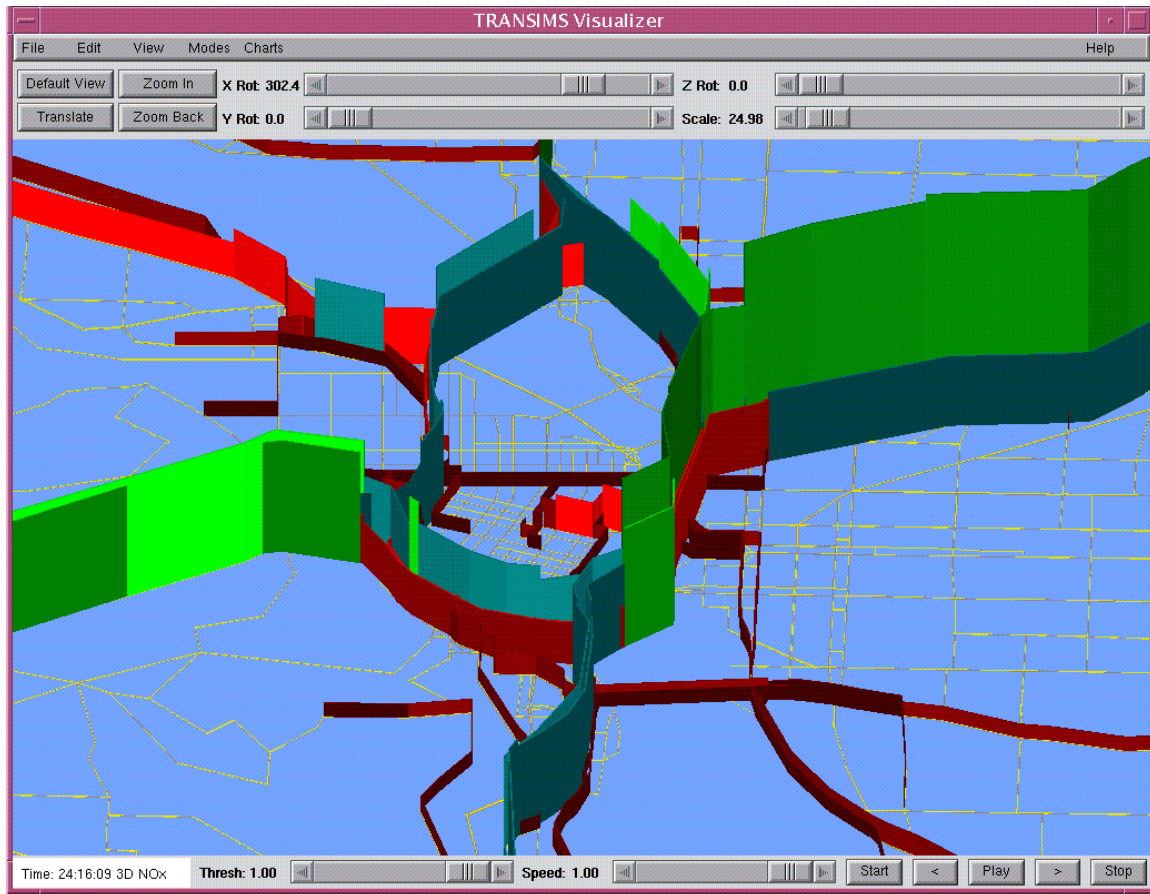


Figure 36: The Output Visualizer displaying cumulative plan data.

Free the memory used for the variable size box data by selecting **File→Close Variable Size Box**.

6.3.2.13 Viewing Constant Size Box Data

At this time, there is no utility to convert summary output data to the file format readable by the Output Visualizer. Read in constant size box data by selecting the **File→Open Summary Data**. The text in the status bar will change to *Reading in Summary Data...* while the file is being read and will return to *Ready...* once completed. Change the background color to white by selecting **Edit→Background Color** and selecting White from the pulldown menu of the Color Selector dialog box that appears; then click [OK]. The display is similar to the one shown in Figure 37.

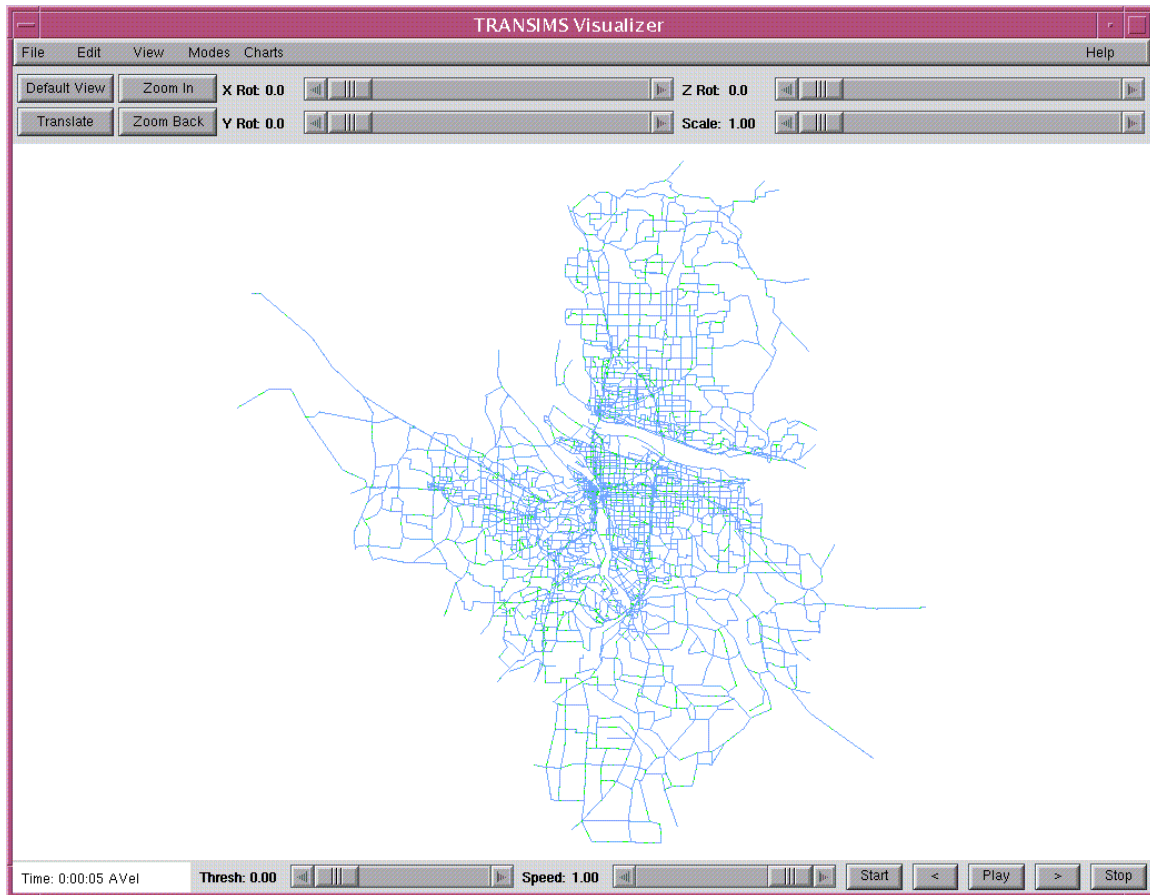


Figure 37: The Output Visualizer displaying average velocity summary data.

As is the case with all the other types of data, animated displays can be started by clicking on the left mouse button within the viewing area and stopped by right mouse clicking. The status bar shows the current timestep being shown and the data type name (Avel in this case).

6.3.2.14 Varying the Colormap Thresholds

In some cases, it may be desirable to determine whether the data value of one link is higher than another, but because of the colormap, the two links will have the same color. Changing the colormap into a more finely grained colormap with more colors can alleviate this problem, but there is an alternative and more convenient way. By cycling the color map, that is by increasing the minimum value color, the links on the display change colors. The link with the higher data value associated with it will change colors before the other link of the same initial color. This cycling of colormaps is accomplished by using the “Thresh” slider (the Thresh slider is visible only when animation mode is switched on).

6.3.2.15 Retrieving Information about a constant length box

Clicking on the center of a constant length box with the MIDDLE mouse button returns information about the box. When you have clicked on a point within 5.0 meters of the center of a

box, information is printed in the text window from which you started the Output Visualizer. A typical output would be:

Found Box 34656 SumV 31.292 Count 2 Cap 33 Link 9059
Avg Velocity: 15.65 Density 0.06 LLx 2330099.75 URx 2330078.00 LLy 208370.12 URy
208289.55
Box Length: 82.80 Width: 10.54 Cells (Est): 33

SumV indicates the sum of vehicle velocities passing through the box within the latest time period, and *Count* indicates the number of vehicles passing through the box during the same time period. *Cap* is the estimated capacity of vehicles within the selected box and is followed by the link id. The average velocity and density for the box are then given, followed by the lower left x coordinate of the box. The upper right x coordinate of the box is then listed, followed by the lower left y coordinate and upper right y coordinate of the box. The length, width, and capacity of the box are listed.

If a box cannot be identified at the point where you have clicked, the message *Box NOT FOUND at xcoordinate ycoordinate* is displayed.

6.3.3 Menu Functionality

6.3.3.1 File

The File menu provides options (Table 34) for opening and closing data files and networks.

Table 34: File menu functionality.

Option	Description
Open Activities	Reserved for later use in displaying activity data.
Open Network	Reads in the network described in the configuration file. The configuration file must specify all network parameters, the GBL_CELL_LENGTH and the GBL_LANE_WIDTH.
Open Variable Size Box Data	Opens a text based data file with variable box sizes. This is used for emissions and summary data that are not based on constant size boxes. A dialog box is displayed that allows you to select the file. Note: Do not double-click on the file selector; when a directory is selected, the contents of that directory will then be listed. The File Selector Dialog Box is shown in Figure 38.
Open Vehicles	Reads in and displays vehicle evolution data from a binary file that summarizes the vehicle locations, types, velocities, etc. Refer to the File Formats section for additional information on the vehicle evolution file format. A dialog box is displayed that allows you to select the file. Note: Do not double click on the file selector; when a directory is selected, the contents of that directory will then be listed.
Open Populations	Reserved for later use in reading in and displaying population data.
Open Summary Data	Opens a constant length box binary summary data file and displays the data by coloring links a color based on the data. See the File formats section for additional information on the binary summary data file. A dialog box is displayed that allows you to select the file. Note: Do not double click on the file selector; when a directory is selected, the contents of that directory will then be listed.
Close Activities	Reserved for later use in closing the current activity data file.
Close Network	Reserved for later use if it is necessary to change networks without quitting the Output Visualizer.
Close Variable Size Box Data	Deallocates memory used for the current variable box data; the data will no longer be accessible.
Close Vehicles	Deallocates memory used for the current vehicle evolution data; the data is no longer accessible. This option should be selected before reading in a new vehicle evolution file or to recover memory for use in displaying another type of data.
Close Populations	Reserved for later use in deallocating memory used in the current Populations data file.
Close Summary Data	Deallocates memory used for the current summary data; the data will no longer be accessible. This option should be selected before reading in a new summary data file or to recover memory for use in displaying another type of data.
Close Populations	Reserved for later use in deallocating memory used in the current Populations data file.
Status Report	Reserved for a full status report of the currently in use data, viewing transformation, and memory use.
Save View to File	Saves the current viewing window to a sun raster file in the current working directory. A dialog box is displayed that allows you to enter a name for the file.
Exit	Quits the Output Visualizer.

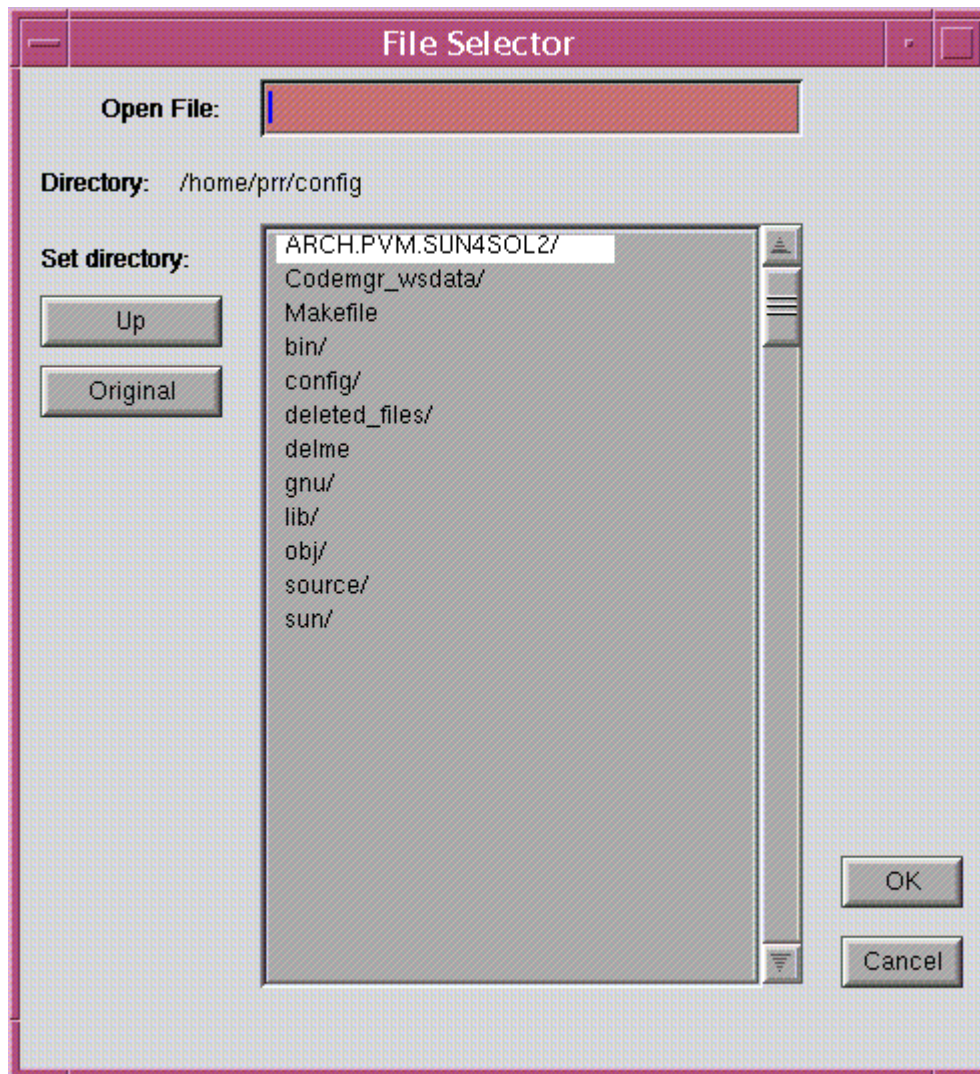


Figure 38: File Selector dialog box.

6.3.3.2 Edit

The Edit menu options (Table 35) allow for finding objects, labeling, and changing background color.

Table 35: Edit menu functionality.

Option	Description
Copy	Reserved for future use in allowing the current viewing area to be pasted into a buffer for use in transferring to other applications.
Find Node	Reserved to find a particular node by its ID Number.
Find Link	Reserved to find a particular link by its ID Number.
Find Parking	Reserved to find a particular Parking Accessory by its ID Number.
Find Box	Reserved to find a particular box by its index (constant length boxes only).
Find Plan	Reserved to find a particular Plan by its ID Number.
Find Vehicle	Finds a given vehicle by its ID Number and colors it in a user-selectable color and a user-selectable point size (when vehicles are displayed as points). This is useful in tracing a single vehicle through the network. The % Box Size numeric input box allows the user to select the size of an area that the chosen vehicle will remain in if Mode→Follow Vehicle is selected. The value input to the % Box Size input box should be between 0.001 and 0.5. The value is the percentage of the window size that the vehicle may stray from the center of the window without re-centering the vehicle in the window. The Find Vehicle Parameters Dialog Box is shown in Figure 39.
Add Text	Allows for setting a user input text label of a given selectable color at a user-selected point within the current viewing area. This is useful in making transparencies for presentations. The Text Note Dialog Box is shown in Figure 40.
Background Color	Allows for the user to change the current background color. The Color Selector dialog box is shown in Figure 41. Note that the Color Selector Dialog Box has its own menu; it is used to select primary colors without having to adjust the sliders. Click [OK] to accept the current background color setting that you have selected.

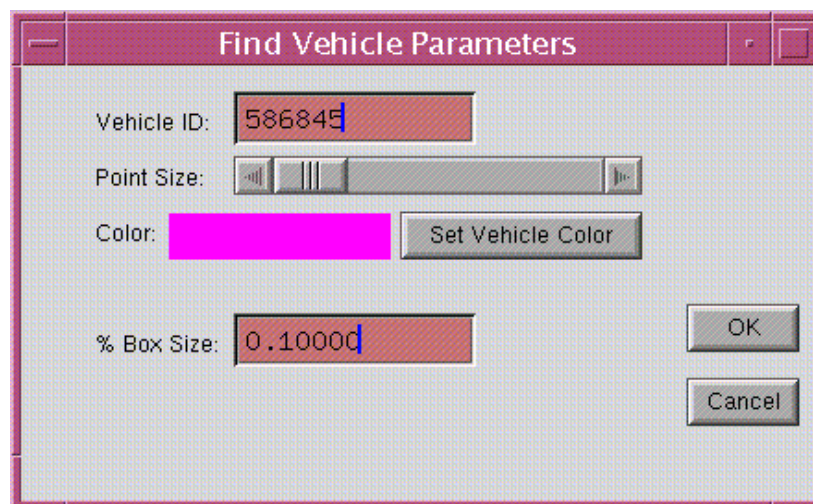


Figure 39: Find Vehicle Parameters dialog box.

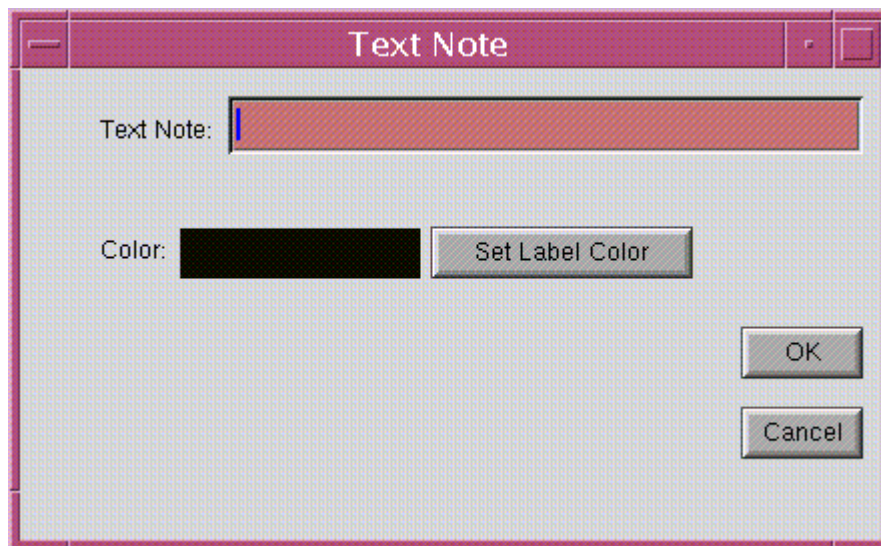


Figure 40: Text Note dialog box.

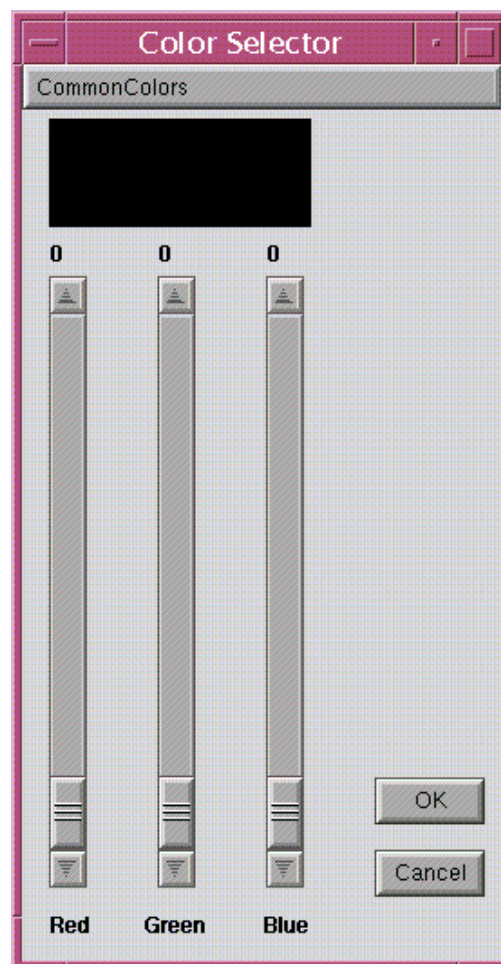


Figure 41: Color Selector dialog box.

6.3.3.3 View

The View menu option (Table 36) allows for the selection of what type of data and/or style the data should be displayed.

Table 36: View menu functionality.

Option	Description
Network	<p>Allows for setting the current network viewing options. The Network Viewing Parameters Dialog Box is displayed as shown in Figure 40.</p> <p>Clicking on the check boxes allows you to select whether or not to display the Nodes, Transit Stops, Parking Accessories, Links, and/or Boxes. Checked boxes indicate that the item will be displayed. The point size can be altered when viewing the Nodes, Transit Stops, and Parking Accessories. All of the colors are user selectable and may be changed by clicking the Set XXX Color buttons. Click [OK] when the viewing options are set to your liking.</p>
Activities	Reserved for future use in setting how activities are displayed.
3D Plans	<p>Link and box based data can be shown as 3-D bar heights, transparency, or mapped to one or two colormaps. This option allows you to select how you would like to view the data in a given column or columns. The data in any given column can be mapped to 3-D bar heights, to transparency or to a specific color. Selecting the 2 Colormaps option allows one column to be represented by mapping the data value from black to another selectable color and another value to be mapped with a complimentary colormap. For example, densities could be mapped with the Black->Red colormap, and velocities could be mapped with the RGB: Right->Left (Blue to Green) colormap. Links where both values are low would then appear as dark blue and yellow where both values are high. The column labels of the current variable size box data file are listed in the Column # - Labels section at the top of the dialog box. The Data Mapping Params dialog box is displayed when View->3D Plans is chosen and is shown in Figure 42.</p> <p>The 3-D scale factor is user selectable and determines the heights of the bars for a given numerical value by multiplying the scale factor by the data value. This number should always be a floating-point number. The transparency min val indicates that all data values falling below this threshold should be drawn fully transparent. The transparency max value indicates that all data values above this threshold should be drawn without transparency. Values that lie between are rendered with a proportional degree of transparency.</p> <p>The Min Val settings for the colormap settings indicate that all data values that fall below this number should be mapped to the first value in the selected colormap. Likewise, the Max Val settings indicate that all data values above this number should be mapped to the last value in the selected colormap.</p> <p>Note: At this time, all colormaps are compiled into the application and are not user changeable. The Set Colormap button is reserved for future use in allowing this new functionality. It should also be noted that each colormap compiled into the application is assigned to a specific column.</p>
Populations	Reserved for future use in setting how population data is displayed.

Option	Description
Vehicles	<p>Allows for setting the current vehicle viewing options. The Vehicle Viewing Parameters Dialog Box is displayed appear as shown in Figure 43.</p> <p>Clicking on the check boxes allows you to select how to color the vehicles.</p> <p>The <i>Same Color Mode</i> colors all of the vehicles in the same user selectable color and the user selectable point size (when vehicles are drawn as points).</p> <p>The <i>Color by Type Mode</i> colors vehicles according to their vehicle type and also renders buses larger than standard vehicle types. Buses are rendered in orange, and the rear section is rendered in purple according to how many passengers are currently in the bus. The point size is also user selectable.</p> <p>The <i>Color by Passengers Mode</i> colors vehicles according to the number of passengers in the vehicle.</p> <p>The <i>Color by Velocity Mode</i> colors vehicles according to their current velocity by a given colormap that is user selectable.</p> <p>The <i>Color by Random Colors Mode</i> colors vehicles according to their vehicle id by a wide range of colors.</p> <p>The vehicles are drawn in three dimensions if the 3-D Vehicles radio box is checked (it will turn yellow when it is checked).</p> <p>Click [OK] when you are satisfied with the current vehicle coloring mode.</p>
Lane Dividers	Toggles whether the lane dividers, drawn as dotted lines, will be shown or not. This will probably be moved to the Network Viewing Parameters dialog box.
Next Data Column	Increments the column number to be shown when viewing the variable size box data in 2-D mode.
Boxes	Reserved for future use.
Densities	Displays densities when constant length box summary data is being drawn.
Speeds	Displays speed data when constant length box summary data is being drawn.
3-D Density/Speed	Reserved for future use in showing constant length box summary data in 3-D.
Labels	Toggles whether Labels should be drawn or not.
Axes	Reserved for future use in setting whether the Axes should be drawn or not. Not implemented at this time.

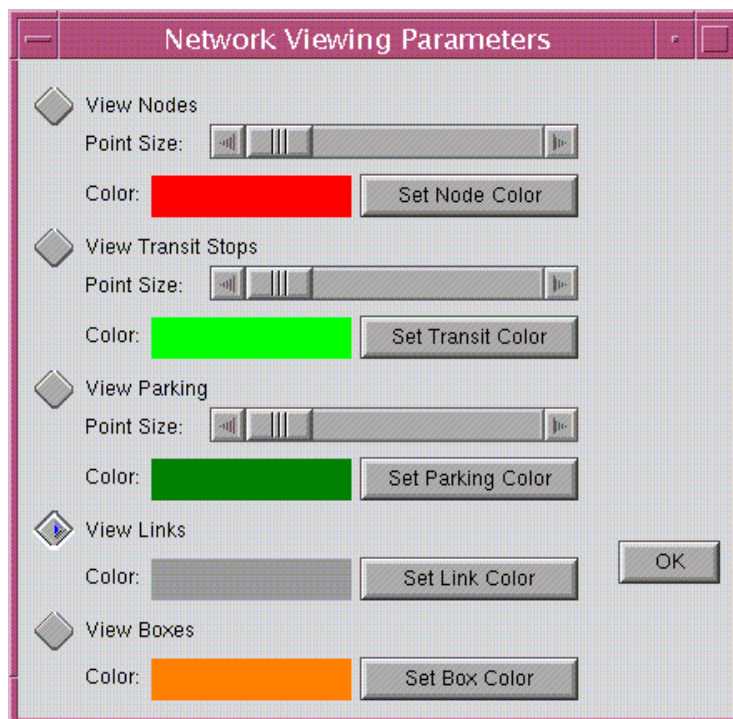


Figure 42: Network Viewing Parameters dialog box.

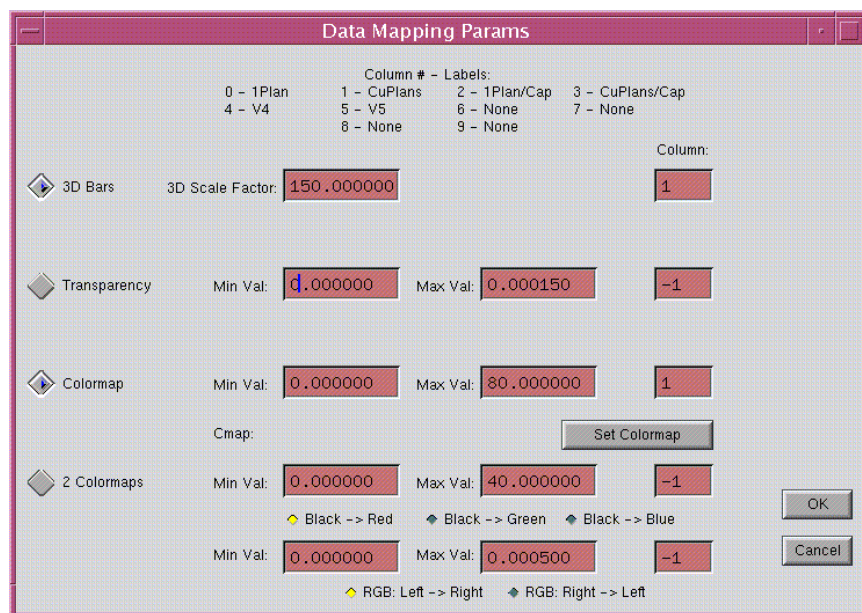


Figure 43: Data Mapping Params dialog box.

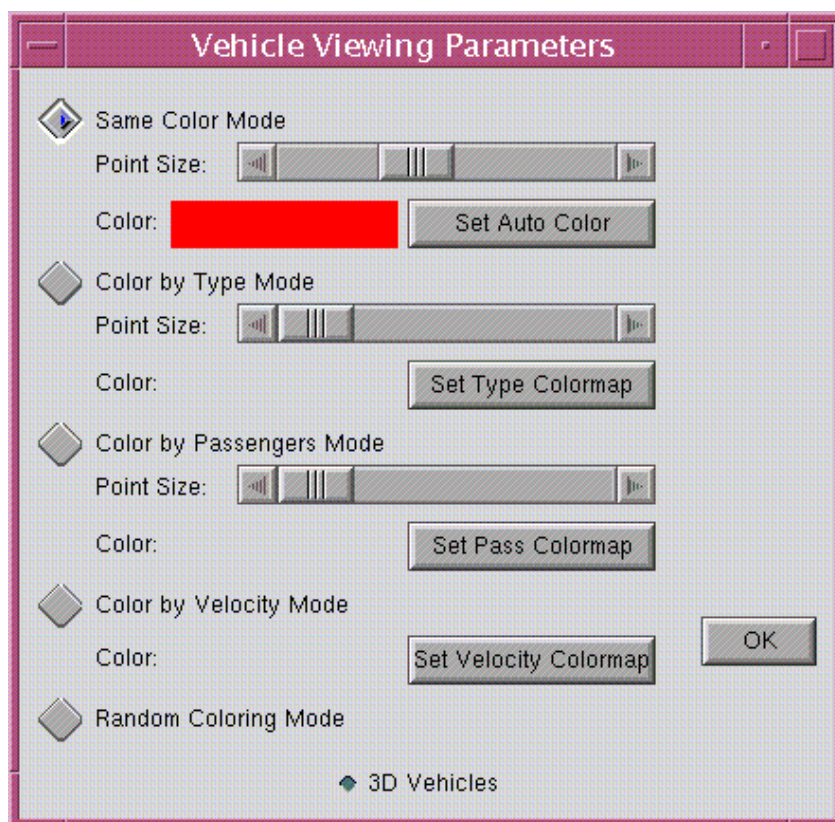


Figure 44: Vehicle Viewing Parameters dialog box.

6.3.3.4 Modes

The Modes menu option (Table 37) allows for the selection of various modes such as whether to use the lighting model, overlay mode, 3-D or 2-D network, etc.

Table 37: Modes menu functionality.

Option	Description
Animation	Allows for the display of buttons in the status bar that are helpful in controlling the speed of animation and timestep positioning.
Still	Hides the display of the buttons and sliders displayed by Modes→Animation.
Lights	Toggles the lighting model on and off. When viewing in 3-D mode, the lights should be switched on in order to correctly render the faces of 3-D polygons.
Antialiasing	Toggles whether antialiasing is done or not. Antialiasing thickens lines and reduces the effect of <i>stairstepping</i> in diagonally drawn lines.
Shading	Toggles whether polygons should be drawn as flat shaded polygons or smooth shaded polygons. Note that smooth shaded polygons will still appear to be flat if normals are not given for each vertex or if all the normals are the same.
Follow Vehicle	Translates the current view, if necessary, making sure that the vehicle chosen from Edit→Find Vehicle is always visible.
Overlay	Toggles whether the overlay mode is <i>on</i> or <i>off</i> . If overlay mode is <i>on</i> , the network is not redrawn for each frame, the pixels of the viewing area are transferred from an area in memory, which was saved after the network was drawn with the current viewing transformations. This saves a large amount of time in drawing complex scenes and makes high frame rate animation possible. However, at this time ,overlay mode has a problem when the lighting model is switched on, so it should not be used in conjunction with lights.
3-D Network	Toggles whether the z-axis values are used when drawing the network. This is useful because you may want to display the topography of the network at times and not display the topography when drawing 3-D boxes so that the boxes may be compared more easily.

6.3.3.5 Charts

The Charts menu option (Table 38) allows for the production of various types of graphs. All items in this menu are not implemented at this time and are reserved for future use.

Table 38: Charts menu functionality.

Option	Description
Flow vs. Density	Reserved for future use.
Space/Time	Reserved for future use.
Delay vs. Density	Reserved for future use.
Delay vs. Time	Reserved for future use.
Capacity vs. Time	Reserved for future use.
Density vs. Duration	Reserved for future use.
Population Histograms	Reserved for future use.
Sources	Reserved for future use.
Destinations	Reserved for future use.

6.3.3.6 Help

The Help menu option (Table 39) allows access to the help facility.

Table 39: Help menu functionality.

Option	Description
Index	An index into the help topics. Not implemented at this time.
Search	Allows for searching for keywords in the help system. Not implemented at this time.
About	Displays a dialog box about the current version of the Output Visualizer. Not implemented at this time.

6.4 Current Colormap Settings

Currently, the colormaps are compiled into the Output Visualizer and are not changeable. However, the colormaps can be scaled to different value ranges in 3-D viewing mode by changing the *Min val* and *Max Val* parameters. The current colormaps are described below by the following format:

- Minimum Value and Maximum Value for the colormap and name or column
- Maximum value for the first color and the color,
- Maximum value for the second color and the color,

```
Min: 0.0 Max: 37.5 Summary Velocity Map
1.0 Dark Red
3.0 Light Red
15.0 Background Blue
30.0 Light Green
35.0 Dark Green
```

```
Min: 0.0 Max: 1.0 Summary Density
0.1 Background Blue
0.2 Light Green
0.3 Orange
0.5 Light Red
1.0 Dark Red
```

```
Min: 0.0 Max 80.0 Emissions Velocity Map or Column 0
1.0 White
20.0 Dark Green
40.0 Yellow
60.0 Orange
80.0 Light Red
```

```
Min: 0.0 Max: 70000.0 Emissions Nitrogen Oxide Map or Column 1
200.0 White
20000.0 Dark Green
30000.0 Yellow
50000.0 Orange
70000.0 Light Red
```

```
Min: 0.0 Max: 1200.0 Emissions Carbon Monoxide Map or Column 2
```


1.0 White
300.0 Dark Green
600.0 Yellow
900.0 Orange
1200.0 Light Red

Min: 0.0 Max: 140000.0 Emissions Hydrocarbons Map or Column 3
200.0 White
20000.0 Dark Green
60000.0 Yellow
100000.0 Orange
140000.0 Light Red

Min: 0.0 Max: 4000.0 Emissions Fuel Economy Map or Column 4
20.0 White
1000.0 Dark Green
2000.0 Yellow
3000.0 Orange
4000.0 Light Red

Min: 0.0 Max: 14000.0 Emissions Flux Map or Column 5
200.0 White
2000.0 Dark Green
6000.0 Yellow
10000.0 Orange
14000.0 Light Red

Min: 0.0 Max: 26000.0 Unused Map or Column 6
5000.0 Dark Red
10000.0 Light Red
15000.0 Teal
20000.0 Light Green
25000.0 Dark Green

Min: 0.0 Max: 18.0 Unused Map or Column 7
3.6 Dark Green
7.2 Light Green
10.8 Teal
14.4 Light Red
18.0 Dark Red

Min: 0.0 Max: 18.0 Unused Map or Column 8
3.6 Dark Green
7.2 Light Green
10.8 Teal
14.4 Light Red
18.0 Dark Red

Min: 0.0 Max: 18.0 Unused Map or Column 9
3.6 Dark Green
7.2 Light Green
10.8 Teal
14.4 Light Red
18.0 Dark Red

Min: 0.0 Max: 255.0 Vehicles by Type Map 12
1.0 Dark Green - Walk
2.0 Light Green - Auto
3.0 Teal - Truck
4.0 Light Red - Bicycle

```
5.0 Dark Red - Taxi
6.0 Orange - Bus
7.0 Black - Trolley
8.0 Blue - Streetcar
9.0 White - Light Rail
10.0 Yellow - Rapid Rail
11.0 Purple - Regional Rail
255.0 Background Blue - Unknown type
```

Min: 0.0 Max: 255.0 Vehicles by Passengers Map 13

```
1.0 Dark Green - 0 passengers
2.0 Light Green - 1
3.0 Teal - 2
4.0 Light Red - 3
5.0 Dark Red - 4
6.0 Orange - 5
7.0 Black - 6
8.0 Blue - 7
9.0 White - 8
11.0 Yellow - 10
26.0 Purple - 25
51.0 Background Blue - 50
101.0 Light Grey - 100
201.0 Grey - 200
255.0 Dark Grey - 255
```

Min: 0.0 Max: 37.5 Vehicles by Velocity Map 14

```
1.0 Dark Red
3.0 Light Red
15.0 Background Blue
30.0 Light Green
35.0 Dark Green
```

6.5 Utility Programs

6.5.1 convcars

The *convcars* program converts the vehicles from the Traffic Microsimulator text output into the binary vehicle position format shown in the file format section. The command to execute the *convcars* program is

```
convcars config inputfilename outputfilename
```

or, optionally,

```
convcars config
```

which will take the IOC2 output file name from the configuration file as the input filename and write to the same filename with the suffix *.bin* added for the output filename.

Using *vehtobin* to convert files is better because it is much faster, uses less memory, and does not care about the ordering of the columns in the vehicle snapshot input file. Use *convcars* only when the following columns are the only columns (and in this order) in the vehicle snapshot file:

```
DISTANCE  LANE  LINK  NODE  TIME  VEHICLE  VELOCITY  PASSENGERS  VEHTYPE
```

6.5.2 vehtobin

The *vehtobin* program converts ioc2 text format to the binary format required by the TRANSIMS Output Visualizer. Usage is as follows:

```
vehtobin inputfilename outputfilename
```

This utility is provided as a quicker alternative to *convcars* in translating to the binary vehicle file format.

6.5.3 Plan2BoxSummary

6.5.3.1 Overview

The purpose of this utility is to prepare a TRANSIMS plan file for display in the Output Visualizer as variable size box data. The utility prepares several different views of the plans, one in each column of the output file. This permits easy viewing of all plans that start during a particular time interval, cumulative plans (accumulated across time intervals), and cumulative plans normalized by the capacity of each link. The plans can either be displayed by individual traveler or summarized over all travelers.

6.5.3.2 Usage

```
Plan2BoxSummary [-f ] [-s startTime] [-e endTime] [-i incTime] [-u userFieldValue] [-t travId]* [-r travIdFile] [-h ] <configFile>  
<planFile>
```

incTime is in seconds. All other times are in seconds since midnight.

Options:

- f plans will be written one at a time, indexed by traveler id.
- s only legs that start on or after this time are written (default 0).
- e only legs that end (as estimated in plan) on or before this time are written (default 86400).
- i bin plans into *incTime* second long bins. (Plan file must be sorted by time.)
- u only plans with the specified value in the user field will be written.
- t only the specified traveler IDs will be written. (The -t option must appear before each ID.)
- r specifies a file of white space-separated IDs to use in addition to any specified by -t.
- h gives this message and exits.

The configuration file and plan file names are both required and must appear in the order shown.

Output is placed in a file named after the plan file argument with the extension *.boxes*.

Example:

```
% $TRANSIMS_HOME/bin/Plan2BoxSummary -i 900 config plans  
Initializing network ... done  
Box summaries written in file plans.boxes
```

```
% head plans.bboxes
TIME link node distance length Single_Plan Cumul_Plans Cap_Normed
12150 1 1 750 750 1864 1864 0.266286
12150 2 2 750 750 1864 1864 0.266286
12150 3 3 750 750 1860 1860 0.265714
12150 4 4 750 750 1860 1860 0.265714
12150 5 5 750 750 1876 1876 0.268
12150 6 6 750 750 1876 1876 0.268
12150 7 7 750 750 1901 1901 0.271571
12150 8 8 750 750 1901 1901 0.271571
12150 9 9 750 750 1890 1890 0.27
```

```
% $TRANSIMS_HOME/bin/Plan2BoxSummary -f -i 600 config plans
Initializing network ... done
Box summaries written in file plans.bboxes
% head plans.bboxes
```

```
TIME link node distance length Single_Plan Cumul_Plans Cap_Normed
167 1 1 750 750 2 2 0.000285714
167 2 2 750 750 2 2 0.000285714
167 3 3 750 750 2 2 0.000285714
167 4 4 750 750 2 2 0.000285714
167 5 5 750 750 2 2 0.000285714
167 6 6 750 750 2 2 0.000285714
167 7 7 750 750 2 2 0.000285714
167 8 8 750 750 2 2 0.000285714
167 9 9 750 750 2 2 0.000285714
```

6.5.3.3 Configuration Keys

The configuration file should contain all the NET_ keys.

Note that the plan file name is specified on the command line, and not taken from the *PLAN_FILE* variable in the configuration file.

6.5.3.4 Troubleshooting

When the -f flag is used, the Output Visualizer interprets the traveler ID as a time. To reconstruct the traveler ID, convert the time into a number of seconds past midnight. Currently, only vehicle driver legs are handled.

This utility has not yet been integrated with the file indexing system.

The plan file should be sorted by departure time or by traveler id.

6.6 Troubleshooting

Problem 6.5.1

Vis not working, returns something like the following error message:

```
X Error of failed request: BadMatch (invalid parameter attributes)
Major opcode of failed request: 1 (X_Create_Window)
Serial number of failed request: 21
Current serial number in output stream: 23
```

Solution 6.5.1

A part of the X server is functioning improperly, reboot the machine to solve the problem. This problem has occurred only on Sun Workstations with Solaris to date.

Problem 6.5.2

A menu appears when I clicked on something else.

Solution 6.5.2

This problem shows up frequently on Sun Workstations with Solaris, it is however not a disastrous problem because clicking on the menu bar area where a menu is not located almost always clears up the problem within a few mouse clicks.

Problem 6.5.3

A slider has stopped working.

Solution 6.5.3

This happens after the window has been resized after a data set has been loaded and appears on both Linux and Solaris systems. With Solaris systems, the problem will eventually go away with subsequent resizing of the window; however, this may take more than twenty resizings of the window. On Linux systems, the problem is only apparent based on where the slider is clicked—if you click on the upper edge of the thumb control, the slider will work. An easy workaround for this problem is to only resize the window prior to loading data sets.

Problem 6.5.4

I click on the Play button and nothing happens.

Solution 6.5.4

When a window has focus (the mouse pointer is within it), processing in other windows halts. Since you have clicked in a window other than the viewing window, the viewing window will not be updated until the current window loses focus. Consequently, this diminishes the capabilities of the Play and Stop buttons. As a workaround, click the left mouse button in the viewing window to start animating (Play button functionality), and click the right mouse button in the viewing window to stop animating (the Stop button functionality). Another problem that stems from clicking on the Play button is that the timestep being displayed seems to jump ahead—skipping many timesteps.